VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Multiwebhostingová platforma

Multiwebhosting platform

2012 **Bc. Václav Duc**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Václav Duc**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Multiwebhostingová platforma
Multiwebhosting Platform

Zásady pro vypracování:

Cílem práce je navrhnout a realizovat platformu, která by byla vhodná pro provoz multiwebhostingu. Uvedené řešení musí být založeno na virtualizaci systémových zdrojů, kdy každému zákazníkovi jsou předem přiděleny minimální a maximální zdroje serveru, které může pro multiwebhosting využívat. Řešení dále musí obsahovat nástroje pro správu multiwebhostingových účtů, včetně plné automatizace procesu vytváření a správy účtů.

1. Seznamte se s aktuální problematikou v oblasti virtualizace. Vypracujte přehled současných virtualizačních řešení z oblasti OpenSource na platformě GNU/Linux.
2. Proveďte otestování jednotlivých virtualizačních řešení a vyberte vhodný typ virtualizace pro provoz multiwebhostingu.
3. Proveďte volbu hostovaného operačního systému a jeho přizpůsobení pro provoz v kontejnerovém prostředí virtuálního stroje.
4. Seznamte se s clusterovymi souborovými systémy OCFS 2 a GFS 2. Proveďte otestování těchto souborových systémů a vyberte souborový systém, který bude pro řešení problému vhodný.
5. Navrhněte a realizujte na technických prostředcích zadavatele takovou infrastrukturu, která bude odolná proti výpadku aplikačního a storage serveru.
6. Proveďte technickou a softwarovou implementaci navržené multiwebhostingové platformy.
7. Při simulované zátěži otestujte navržené řešení.

Seznam doporučené odborné literatury:

Podle pokynů vedoucího diplomové práce.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Libor Holub**

Konzultant diplomové práce: Ing. Petr Olivka

Datum zadání: 18.11.2011

Datum odevzdání: 04.05.2012

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne: 4. 5. 2012 Podpis autora:

Prohlášení firmy

Souhlasím se zveřejněním této diplomové práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v bakalářských/magisterských programech VŠB-TU Ostrava.

V Ostravě dne: 4. 5. 2012 Podpis autora:

Libor Holub
NETtip
Krakovská 5
700 30 Ostrava 3
tel./fax: 596 716 847
info@katalogtip.cz
IČO 70311919 DIČ CZ7901045581

Poděkování

Rád bych poděkoval panu Ing. Liboru Holubovi Ph.D. za vloženou důvěrou, příležitost a poskytnutí technických i duševních prostředků pro realizaci této diplomové práce.

Poznámka

Práce byla vypracována a realizována ve firmě Libor Holub – NETtip, kterou budu dále v textu uvádět pod obchodní značkou SvetHostingu.cz nebo jako zadavatele.

Abstrakt

Diplomovou práci jsem vypracoval u firmy SvetHostingu.cz, kde jsem již dříve formou své bakalářské práce realizoval VPS řešení na virtualizační technologii Xen. Tato diplomová práce se zabývá návrhem, implementací a testováním multiwebhostingové platformy. V práci je uveden přehled současných open source virtualizačních řešení na platformě operačních systémů GNU/Linux a výběr nejvhodnějšího z nich pro realizaci této platformy. Pro zvolené virtualizační řešení je dále popsána příprava a vhodné přizpůsobení hostovaného operačního systému tak, aby co nejlépe vyhovoval požadavkům a prostředí firmy a svou funkčností svým potenciálním zákazníkům. Na řešení jsou kladeny mimo jiné také požadavky ze strany vysoké dostupnosti služby, tudíž se v práci zabývám také implementací vhodné clusterové technologie a clusterových souborových systémů. Práce dále obsahuje popis konečné praktické realizace na fyzických prostředcích firmy a uvedení multiwebhostingové platformy do produkčního prostředí. Součástí praktické realizace je i implementace monitorování platformy.

Abstract

The diploma thesis has been created at SvetHostingu.cz company, where as a Bachelor's thesis I implemented a VPS solution based on a virtualization technology Xen. This diploma thesis is focused on design, implementation and testing of a multiwebhosting platform. In the thesis there is an overview of current open source virtualization solutions for GNU/Linux operating system platform and a selection of the most convenient one for the multiwebhosting platform. For the choosen virtualization solution is described preparation and customization of the host operating system to fit the best requirements of the company and with its functionality to potential customers. An additional requirement is on high availability of the service, thus I deal also with implementation of the cluster technology and cluster file systems. The thesis further describes practical realization on real hardware in the company and launch of the platform into production environment. The practical realization also includes implementation of multiwebhosting monitoring.

Klíčová slova

Hosting, Webhosting, Multiwebhosting, virtualizace na úrovni operačního systému, Gentoo GNU/Linux, OpenVZ, reverzní proxy server, cluster, DRBD, OCFS2, GFS2.

Keywords

Hosting, Webhosting, Multiwebhosting, operating system – level virtualization, Gentoo GNU/Linux, OpenVZ, reverse proxy server, cluster, DRBD, OCFS2, GFS2.

Seznam použitých symbolů a zkratek

ACL	Access Control List
CT	Container
CTID	Container Identification
DRBD	Distributed Replicated Block Device
GFS2	Global File System 2
GNU	GNU's Not Unix
HA	High Availability
HN	Hardware Node
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IP	Internet Protocol
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
KVM	Kernel - based Virtual Machine
LVM	Logical Volume Management
MTU	Maximum Transmission Unit
OCFS2	Oracle Cluster File System 2
OS	Operační Systém
RAM	Random Access Memory
VE	Virtual Environment
VMM	Virtual Machine Monitor
VPS	Virtual Private Server

Seznam použitých obrázků

Seznam použitých tabulek

Obsah

1 Úvod

Cílem této práce je popsat kompletní návrh a implementaci multiwebhostingové platformy pro komerční využití firmou zadavatele.

V dnešní době je elektronická prezentace ať již pro soukromé nebo firemní účely důležitější, než kdy předtím. Jedinci chtějí publikovat své myšlenky a zážitky, firmy potřebují snadno dostupné elektronické katalogy, obchody nebo jen prezentaci své činnosti. Rozmach internetu a především pak služby webových stránek a elektronické pošty přinesl na trh nové odvětví – hostingové služby. Pod pojmem hostingová služba si dnes nejčastěji představíme prostor pro webové stránky, který může být zobrazen skrze systém www, elektronickou poštu a databázové služby. Základem každého hostingu je doménové jméno. Užíváme jej pro přístup ke stránkám a je součástí e-mailové adresy. Zjednodušeně lze říci, že co doménové jméno to hostingová služba.

Dnes má své doménové jméno téměř každý jedinec, větší firmy pak běžně mají domén více, ať již dle svých produktů nebo odvětví, kterým se věnují. S tímto trendem již není hosting jednoduchá služba pro vyvolené. Stoupá poptávka a také konkurence. Zákazník jde tam, kde je služba kvalitnější, poskytuje více nastavení a hlavně je levnější. Navíc čím dál častěji tvůrci stránek na míru i poskytovatelé webových šablon poskytují svým zákazníkům kompletní servis nejen naprogramováním webové aplikace, ale také zřízením a provozem hostingových služeb. Málokterý tvůrce (i v případě větší firmy na vývoj internetových prezentací) si ale může dovolit zajišťovat kompletní hostingový provoz, a tak více či méně spolupracuje s některým z poskytovatelů hostingů. Tato firma pak po poskytovateli hostingových služeb požaduje zpravidla velké množství jednotlivých hostingových účtů pro domény svých zákazníků, které se navíc často značně liší co do konfigurace hostingového prostředí. Samozřejmě, že pak není pro tuto firmu a zákazníka ekonomické, aby pro každou jednu doménu byl zakoupen samostatný hosting. Toto ale není případ pouze firem na tvorbu webových stránek. Firma SvetHostingu.cz se běžně setkává i s dalšími zákazníky, kteří z různých důvodů požadují velké množství hostingových účtů a jejich specifické konfigurace.

Běžný webhosting (nazývá se také jako **sdílený webhosting**) pro webové stránky funguje na webovém serveru, který je sdílen pro velké množství domén různých zákazníků, často se jedná až o tisíce. Verze webového protokolu HTTP 1.1 [2] a Name - based Virtual Host rozšíření webového serveru Apache [3] umožňuje, aby bylo takové množství domén svázáno s jedním webovým serverem s jednou nebo pár veřejnými IP adresami. Takže nedochází k plýtvání IPv4 adres, kterých je již tak málo. Takové řešení je velice snadné na konfiguraci a levné na provoz. Jeden server je pak schopen obsloužit stovky až tisíce domén, dle vytížení zákazníky. Toto sdílené prostředí má však několik nevýhod. Menší konfigurovatelnost, slabá izolace a tím i zabezpečení jednotlivých domén a také nemožnost jednoduše přidělovat systémové zdroje jednotlivým doménám. Navíc má-li zákazník domén více, jistě by bylo vhodnější, aby všechny sdílely konkrétní nastavení systémových zdrojů. O dostupné prostředky se pak rozdělí a zbylé místo může špičkově využít libovolná z nich. V případě mnoha domén na sdíleném webovém serveru je pak takové nastavení prakticky nemožné a snadno může i dojít k přetížení serveru jedním nebo pár zákazníky.

Vyhrazený webhosting ať již formou samostatného fyzického nebo virtuálního serveru (VPS) pomůže vyřešit problém s izolací, systémovými zdroji i konfigurovatelností. Zákazník si pak může v pronajatém prostoru prakticky libovolně přizpůsobit softwarové prostředí – má absolutní konfigurovatelnost. Toto řešení je však náročné na správu a znalosti. Zákazník pak musí často přikupovat navíc ještě technickou podporu, nemluvě o tom, že striktní vyhrazení systémových zdrojů na serveru nebo celého fyzického serveru je záležitost velice nákladná.

Proto se často hledají způsoby, jak dát dohromady výhody sdíleného a vyhrazeného webhostingového řešení. Výsledná služba by tedy měla být dobře nebo absolutně konfigurovatelná, zákazníci by měli být silně izolováni a měla by být možnost jim přidělovat rozličné systémové zdroje. Tyto zdroje však ale nesmí být striktně vyhrazeny jako v případě vyhrazeného webhostingu, ale budou zákaznickým prostředím zaplňovány a uvolňovány dle potřeby. Služba by také měla klást minimální nároky na konfiguraci a znalosti ze strany zákazníka a také na čas technické správy ze strany poskytovatele služby. Řešení musí být efektivní co do použitých hardwarových zdrojů. Tyto benefity má poskytnout **multiwebhosting** na vhodném virtualizačním řešení.

SvetHostingu.cz využívá výhradně open source řešení na poli operačních systémů i softwarového prostředí. Nejprve jsem tedy musel projít dostupná otevřená virtualizační řešení pro platformu operačních systémů GNU/Linux. Na základě jejich prostudování jsem se mohl rozhodnout, které z řešení bude nejvhodnější pro realizaci multiwebhostingové platformy. Pro toto virtualizační řešení jsem pak musel navrhnout strukturu virtualizované jednotky – samotného virtuálního stroje, který bude sloužit jako prostředí pro zákazníka. Jelikož výpadek služby, která je poskytována stovkám a tisícům zákazníků by způsobil značné škody, musel jsem se zabývat také řešením vysoké dostupnosti multiwebhostingové platformy. S tím pak souvisí clusterové řešení DRBD a clusterové souborové systémy OCFS2 a GFS2. S vhodně navrženým clusterem a virtuálním prostředím jsem mohl začít multiwebhostingovou platformu implementovat na fyzických prostředcích firmy. Implementace obsahovala jednak hardwarovovou konfiguraci serverů realizujících službu a především potom softwarovou implementaci multiwebhostingu. Ta spočívala v přípravě hostitelských operačních systémů na navrženém clusteru, zprovoznění vybraného virtuálního prostředí a připraveného virtuálního stroje pro zákazníky. S funkčním multiwebhostingovým prostředím jsem musel dále zautomatizovat administrační procesy, které budou nad touto platformou vykonávány. Dle požadavků zadavatele je totiž proces od objednávky zákazníkem až po spuštění multiwebhostingové služby plně automatický, bez zásahu obsluhy ze strany firmy. Automaticky se dějí také další akce, jako je rušení nebo modifikace služby. S funkční zautomatizovanou strukturou multiwebhostingu jsem musel ještě vyřešit jeho produkční otestování a také monitorování. To bude sloužit nejen jako statistika ze strany firmy zadavatele, ale také pro zákazníky samotné, kteří poté budou mít zpřístupněny grafy zátěže systémových zdrojů, případně další statistiky. Ze strany firmy byly také kladeny požadavky na vytvoření instalační a konfigurační dokumentace všech implementovaných částí. Proto v této práci uvádím na příslušných místech i výseky z těchto instalací, případně konfigurací.

2 Open Source Virtualizace na platformě GNU/Linux

Na poli open source virtualizačních řešení pro operační systémy typu GNU/Linux existuje v současnosti několik dobře známých a používaných virtualizačních řešení. Virtualizace samotná může být prováděna několika technikami. Některá z virtualizačních řešení pak podporují těchto technik více.

Pojem virtualizace bude v této práci zmíněn mnohokrát, a proto než zde blíže popíšu techniky virtualizace a konkrétní virtualizační řešení, rád bych definoval některé důležité pojmy ze světa virtualizace.

- **Nativní běh** – operační systém spuštěný přímo na fyzickém počítači bez přítomnosti virtualizace.
- **Hostitel** (pozor, v angličtině se jedná o slovo *Host*) - jedná se o fyzický počítač s operačním systémem, který spouští a spravuje virtualizované stroje.
- **Host** (pozor, v angličtině se jedná o slovo *Guest*) – virtualizovaný počítač s operačním systémem, izolovaná jednotka od ostatních hostů.
- **Hypervizor** (také VMM – Virtual Machine Monitor) – řídí přístup virtualizovaných počítačů k hardwaru. Může být dvojího typu, *nativní hypervizor* – běží přímo nad hostitelským hardwarem a *hostovaný hypervizor* – běží v rámci prostředí hostitelského operačního systému.
- **Hardwarová virtualizace** - poskytuje prostředí s plně nebo částešně virtualizovaným hardwarem. Operační systém uvnitř této virtualizace má tak k dispozici kompletní počítač. Zahrnuje plnou virtualizaci, paravirtualizaci a emulaci.
- **Softwarová virtualizace** – poskytuje softwarové prostředí, které je nějakým způsobem izolováno od hostitele a dalších hostů, ale nemá k dispozici svůj virtualizovaný hardware. Toto prostředí je pak většinou velmi úzce svázáno s hostitelem a jím poskytovaným rozhraním. Zahrnuje virtualizaci na úrovni operačního systému a aplikační virtualizaci.

2.1 Techniky virtualizace

Pro techniky virtualizace neexistuje obecné rozdělení. Já zde popisuji ty typy virtualizace, které využívají nejběžnější virtualizační řešení dostupné pod GNU/Linuxem, a které jsou zajímavé z hlediska vyhrazeného hostingu a multiwebhostingu.

2.1.1 Emulace

Emulace je typ virtualizace, který vytváří softwarově kompletní virtuální prostředí, včetně procesoru. Lze tedy například na architektuře x86 emulovat počítač s procesorem MIPS. Tento typ

virtualizace má velice slabý výkon a nehodí se pro produkční použití, jelikož i v případě provozu x86 procesoru nad x86 procesorem dochází k jeho emulaci.

Výhody:

- Dokáže softwarově napodobit libovolnou architekturu a periferie.
- Neklade si žádné požadavky na jádro operačního systému nebo podporu v procesoru.

Nevýhody:

- Díky softwarové emulaci veškerého hardwaru včetně procesoru je emulace velice pomalá.

2.1.2 Plná virtualizace

Virtualizovanému operačnímu systému je poskytován kompletní virtualizovaný hardware, kromě procesoru. Host tedy vidí kompletní počítač, přičemž úlohy jsou ale vykonávány nativně na procesoru hostitele. Host má k dispozici vždy stejný druh procesoru jako hostitelský operační systém. Problém jsou privilegované operace, jako je například přístup k hardwaru. V případě, že se host snaží komunikovat s hardwarem, např. pevným diskem, musí být tato komunikace odchycena a ošetřena. Zápis na pevný disk, který má k dispozici host je tedy převeden na zápis na disku hostitele. Taková akce zpomaluje běh virtualizovaného prostředí.

Na x86 architektuře byla do procesorů přidána rozšíření AMD-V (AMD) a VT-x (Intel), která zpřístupňují tzv. hardwarově asistovanou virtualizaci. Ta pomáhá zvýšit výkon plné virtualizace při vykonávání privilegovaných instrukcí.

Výhody:

- K dispozici je kompletní virtuální hardwarové prostředí.
- Host o virtualizaci nic netuší, ani nemusí, je tedy možné virtualizovat libovolný operační systém, který má podporu pro procesor hostitele. Snadno lze tedy spustit proprietární operační systémy typu Microsoft Windows.
- Úlohy využívající výhradně výpočetní čas procesoru běží téměř s výkonem nativního běhu. Ke zpomalení dochází jen při ošetřování privilegovaných operací.

Nevýhody:

- Nutnost ošetřovat privilegované operace. Nízký výkon při práci s hardwarem (nevýkonné I/O).

2.1.3 Paravirtualizace

Paravirtualizace využívá úpravy jádra hostitelského i hostovaného operačního systému tak, aby jisté privilegované operace vyvolané hosty nemusely být zachytávány a složitěji zpracovány, ale

jsou prostřednictvím speciálních volání předávány přímo přes jádro hostitele k hypervizoru a hardwaru.

Výhody:

- Vyšší výkon než u plné virtualizace především pro I/O operace.
- K dispozici je většina hardwarového vybavení. Není problém do virtualizovaných hostů exportovat vybrané periferie hostitele.

Nevýhody:

- Jádra použitých operačních systémů musí být upravena. Nelze tedy použít pro virtualizaci proprietárních operačních systémů jako je Microsoft Windows.
- Lze použít jen jádra, která jsou k tomuto typu virtualizace upravena nebo pro ně existují vhodné patche.

2.1.4 Virtualizace na úrovní operačního systému (kontejnerová virtualizace)

Jedná se o softwarovou virtualizaci. Pro svůj běh využívá upravené jádro hostitelského operačního systému. To vytváří pro virtualizované hosty rozhraní pro přístup ke zdrojům hostitele. Tak se dá snadno na úrovni jádra řídit přístup k paměti, datovému uložišti, síti a dalším systémovým zdrojům. Nedochází tedy k virtualizaci samotného hardwaru, pouze o řízení a limitaci prostředků, které má fyzicky dostupné hostitelský operační systém. Procesy hostů jsou spouštěny v kontextu hostitele, avšak jsou navzájem mezi hosty izolovány. Tím se zajistí oddělení hostů. Nedochází tedy k paralelnímu spuštění více operačních systémů najednou, jak je tomu u plnohodnotných hardwarových virtualizací, byť se host zevnitř tváří jako samostatný operační systém.

V principu se jedná vlastně o zdokonalený unixový *chroot* [4], který je vybaven větší mírou izolace a řízením systémových zdrojů. Procesy tvořící běh hosta jsou uzamčeny v definovaném umístění.

Výhody:

- Vysoký výkon a nízká režie – žádný jiný typ virtualizace nemůže nabídnout vyšší výkon vzhledem k nativnímu běhu. Procesy hostů jsou spouštěny izolovaně v kontextu hostitele. Není emulován žádný hardware pro hosty, nemusí se odchytávat a vyřizovat privilegované operace, využívá se vrstvy jádra pro poskytování systémových zdrojů hostitele. Hardware jako takový je dostupný standardním způsobem pro hostitele a jádro pouze spravuje přístup pro procesy kontejnerů.
- Efektivnější správa paměti - hosté využívají jen tu paměť, kterou potřebují pro běh svých procesů. Není tedy třeba vyhradit určitou velikost operační paměti, která se nastálo ukrojí z operační paměti hostitele tak jako u hardwarových virtualizací.

- Dostupnost hostů pro hostitele – hostitel vidí spuštěné procesy hostů, jelikož běží v jeho kontextu. Také diskový prostor je běžně u této virtualizace snadno dostupný jako adresář, ve kterém je host uzamčen. Hostitel tedy může snadno provádět například zálohování.

Nevýhody:

- Nejedná se o virtualizaci v pravém slova smyslu, host nemá k dispozici ani virtualizovaný hardware, spoléhá se tedy jen na prostředky, které mu dovolí rozhraní v jádře. To může vést například k problémům s pokročilejšími typy síťových zařízení apod.
- Lze použít jen operační systémy stejného druhu, pod GNU/Linuxem tedy lze tímto způsobem virtualizovat pouze GNU/Linux.
- Nutnost používat silně patchované jádro dodávané tvůrci virtualizačního řešení.
- Společné jádro pro hostitele i hosta.

2.2 Virtualizační řešení

Virtualizační řešení poskytují konkrétní implementaci jedné nebo více technik virtualizace. Umožňují tedy již přímo spouštět více operačních systémů pod jedním hostitelem. Běžně sebou každé virtualizační řešení přináší také nástroje pro vytváření a správu virtualizovaných operačních systémů, případně další pokročilé funkce jako je třeba živá migrace, síťování v různých režimech, konverze virtuálních strojů mezi různými řešeními apod.

Zmiňuji zde pouze dlouhodobě aktivně vyvíjená otevřená virtualizační řešení, která mají dobrou dokumentační základnu a jsou určena pro stabilní produkční nasazení. Důležitým kritériem je také požadavek na možnosti nastavení, především pak z hlediska limitace systémových zdrojů, které je pro implementaci multiwebhostingu velice důležité. Ze serverové scény jsem vybral řešení KVM, Xen, Linux-VServer a OpenVZ. VirualBox je pak vybrán jako jediný zástupce určený primárně pro uživatelské použití s GUI, byť obsahuje také řádkové nástroje.

Konkrétním výběrem vhodného virtualizačního řešení pro multiwebhostingovou platformu se zabývám v kapitole 3.

2.2.1 KVM (Kernel - based Virtual Machine)

Virtualizační řešení pro GNU/Linux [5], které má pravděpodobně největší budoucnost, jelikož je přímo součástí Linuxového jádra a je s ním vyvíjeno a pravidelně aktualizováno. KVM je k dispozici v jádrech od verze 2.6.20. KVM podporuje plnou virtualizaci s hardwarově asistovaným rozšířením. Vyžaduje tedy podporu na straně procesoru (rozšíření AMD-V, resp. Intel VT-x). KVM poskytuje také omezenou možnost paravirtualizace, v současnosti se jedná o speciální ovladače pro síťovou kartu a řadič disků. Tyto ovladače umožňují dohnat ztrátu v oblasti I/O operací. Více viz VirtIO [6].

KVM využívá hostovaný hypervizor. Ten se aktivuje načtením příslušného modulu do jádra zavedeného hostitelského operačního systému. Přes dodané nástroje je pak možné spustit hostované operační systémy. Na KVM řešení je možné provozovat prakticky libovolné operační systémy bez jejich úpravy. Bez problémů pod ním fungují GNU/Linuxové distribuce, BSD systémy, Microsoft Windows, Solaris a další.

Standardně jsou dostupné nástroje pro příkazový řádek. KVM je určeno především pro serverové nasazení. Existují však také grafické ovládací panely, viz [7].

2.2.2 Xen

Virtualizační řešení Xen [8] bylo původně vyvíjeno pro čistou paravirtualizaci jako serverové řešení. Ve své verzi 3.0 byla však přidána také podpora režimu plné virtualizace, konkrétně s rozšířením na hardwarově asistovanou virtualizaci. V tomto režimu je tedy nutná podpora procesoru AMD-V, resp. Intel VT-x.

Nejnižší vrstvou nejblíže hardwaru je Xen hypervizor. Xen využívá nativní typ hypervizoru. Ten je po startu počítače zaveden nejdříve a následně spustí hostitelský operační systém. Ten se v terminologii Xenu nazývá privilegovaná doména nebo také doména 0. Tento privilegovaný operační systém pak může skrze nástroje spravovat hostované operační systémy, kterým se říká neprivilegované domény nebo také domény U.

Na Xenu je možné využít jak režim paravirtualizace, tak plnou virtualizaci s hardwarově asistovaným rozšířením. Je tedy možné spouštět téměř libovolný operační systém tak jako u řešení KVM. Pro režim paravirtualizace samozřejmě vyplývá omezení použití pouze na operační systémy s otevřenými jádry, jelikož je nutná jejich modifikace. V současnosti lze v režimu paravirtualizace stabilně provozovat libovolné operační systémy typu GNU/Linux. Existují i postupy pro provoz BSD systémů, ale ty nejsou dostatečně stabilní. Jelikož Xen v režimu paravirtualizace vyžaduje upravená jádra pro hostitele i hosta, bylo nutné dlouhou dobu používat velice zastaralé oficiální jádro 2.6.18 nebo zvláště opatchovaná komunitní jádra, která však ne vždy oplývala stabilitou. Xenu se dlouhou dobu nedařilo oficiálně proniknout do Linuxového jádra, jelikož jeho vývojáři odmítali přijmout kód Xenu údajně z důvodu jeho nečistoty. Jádro verze 2.6.37 a vyšší je již možné využít jako hostitelské pro privilegovanou doménu 0.

S Xenem jsou dodávány standardní řádkové nástroje, které kromě běžné správy hostovaných operačních systémů také obsahují prostředky pro živou migraci hostů mezi hostiteli. Nejpoužívánější grafický ovládací panel je pak Virtual Machine Manager [9] od Red Hatu.

2.2.3 Linux-VServer a OpenVZ

Dvě otevřená řešení, která poskytují virtualizaci na úrovni operačního systému. Obě řešení používají Linuxová jádra, která jsou patchována pro podporu virtualizace. Linux-VServer [10] je

komunitní projekt, který je v současnosti spravován Herbertem Pötzlem. OpenVZ [11] je rovněž komunitní projekt a je podporován a sponzorován společností Parallels [12], které slouží jako základ pro její vlastní virtualizační uzavřené řešení Virtuozzo Containers [13].

Podpora operačních systémů je dána typem virtualizace a ta omezuje toto řešení na distribuce GNU/Linuxu. Verze nástrojů v těchto distribucích pak musí být navíc schopny spolupracovat s verzí jádra, které Linux-VServer a OpenVZ poskytují. Tento typ virtualizace nepoužívá řešení na principu hypervizoru, ale zavádí speciální rozhraní do jádra, které umožňuje virtualizaci hostů.

OpenVZ má tu výhodu, že za ním stojí silná komerční společnost, která jej financuje a časem pro něj uvolňje některé své komerční technologie k použití. Navíc OpenVZ pro své patche umožňující virtualizaci používá pravidelně bezpečnostně udržované stabilní jádro z Red Hat Enterprise Linuxu [15], do kterého jsou také dodávány nejnovější komponenty.

Pro obě řešení jsou základem nástroje na úrovni příkazové řádky, jelikož jsou zamýšleny čistě na serverovou sféru. Pro OpenVZ však existuje velká řada grafických ovládacích panelů [16].

2.2.4 VirtualBox

Software VirtualBox [15] byl původně vyvíjen firmou Innotek, následně byl zakoupen společností Sun Microsystems a tu nedávno převzal softwarový gigant Oracle. Jedná se o virtualizační řešení, jehož hlavním cílem je nabídnout výhody virtualizace obyčejným uživatelům a nemíří na serverovou sféru. Poskytuje tedy v základu příjemné uživatelské prostředí s intuitivním ovládáním. Instalace je rychlá a snadná za účasti grafického instalátoru. Velká část VirtualBoxu je distribuována pod GPL licencí. Výjimku tvoří některé proprietární komponenty jako jsou USB 2.0 ovladače, RDP a PXE bootování. Tyto proprietární ovladače jsou instalovány jako rozšiřující balíček.

VirtualBox poskytuje plnou virtualizaci s podporou rozšíření pro hardwarově asistovanou virtualizaci, podporuje-li hostitelský počítač rozšíření AMD-V, resp. Intel VT-x. Toto rozšíření je možné pro jednotlivé hostované operační systémy nastavit. Není tedy nutný procesor s podporou takového rozšíření jako v případě KVM nebo Xenu v režimu plné virtualizace.

Díky použitému typu virtualizace je možno bez úprav instalovat téměř libovolný operační systém stejně jako v případě KVM, tedy včetně uzavřených operačních systémů firmy Microsoft.

Jak bylo řečeno výše, VirtualBox je určen do uživatelské sféry jako grafická aplikace a také se tak v základním režimu chová. Poskytuje ovšem i řádkový tzv. „headless” mód.

3 Virtualizační řešení pro multiwebhosting

V této kapitole definuji požadavky kladené na multiwebhostingovou platformu a z toho vyplývající kritéria pro výběr virtualizačního řešení. Přinesu zde výsledky výkonových testů vybraných virtualizačních řešení a přehled funkcí, které poskytují. Na základě těchto údajů provedu výběr vhodné vritualizační platformy.

3.1 Stanovení požadavků na multiwebhostinogovou platformu

Jak již bylo zmíněno v úvodu této práce, multiwebhostingová platforma má kombinovat výhody klasického sdíleného webhostingu s vyhrazeným webhostingem.

Firma zadavatele v současnosti provozuje následující typy webhostingu:

Sdílený webhosting

- maximálně 2 domény
- cena řádově desítky korun za měsíc
- sdílené systémové prostředky
- sdílený webový server s globální konfigurací
- bez správcovského přístupu
- zanedbatelná údržba ze strany zákazníka
- integrace do webové administrace poskytované firmou, automatizované administrační procesy

Vyhrazený webhosting*

- neomezené množství domén
- cena řádově několik stovek až tisíce korun (bez ceny za placenou správu) za měsíc
- vyhrazené systémové prostředky
- vyhrazený webový server, plně konfigurovatelný
- správcovský přístup
- nutná dobrá znalost softwarového prostředí i operačního systému
- administrace vlastními prostředky nebo některým z dostupných ovládacích panelů typu ISPconfig nebo Webmin, bez administračních procesů

* *Vyhrazený webhosting je možné realizovat buď fyzickým nebo virtuálním serverem (VPS). Zavedení VPS hostingu jsem pro tohoto zadavatele vypracoval v rámci své bakalářské práce. Viz [1].*

Sdílený webhosting je pro zákazníka limitující v maximálním počtu hostovaných domén. Současný nejvyšší program sdíleného webhostingu umožňuje hostovat max. 2 domény. Dalším limitujícím faktorem je konfigurace webového serveru. Mnoho voleb lze nastavit jen sdíleně nebo není politicky vhodné je na tomto druhu služby poskytovat. Výhodou je nízká cena a prakticky nulová údržba ze strany zákazníka. Ten si jen prostřednictvím FTP a MySQL klientů nahraje data na hostingový prostor a ta „začnou fungovat“. Ke změně dostupných nastavení má pak zákazník k dispozici webovou administraci poskytovanou firmou. Ze strany hostingu je nevýhodné sdílení prostředků systému z hlediska výkonu i bezpečnosti jednotlivých zákazníků.

Vyhrazený webhosting na straně druhé je pro většinu zákazníků zbytečný luxus. Dává naprostou volnost co do instalovaného softwarového prostředí a jeho konfiguraci, ale z hlediska údržby je náročný na znalosti softwarového prostředí i samotného použitého operačního systému. Vyhrazený webhosting je zároveň podstatně dražší oproti sdílenému webhostingu, což se pouze v případě nároků na větší množství domén a specifickou konfiguraci web serveru nemusí vždy vyplatit. Navíc toto může být prodraženo také nutností doplácet si za správcovské služby, jelikož ne každý zákazník rozumí prostředí vyhrazeného serveru. Jelikož je tato služba natolik individuální, není ani principiálně možné ji zavést do pohodlné webové administrace poskytované firmou. Zákazník pak musí používat ovládací panely třetích stran, které však ne vždy fungují tak přesně, jak je od nich vyžadováno a jejich dodatečná úprava je nevyhnutelná.

Dnešní nejčastější zákazník webhostingu potřebuje hostovat větší množství domén se specifickými konfiguracemi web serveru, zároveň ale nepotřebuje mít k dispozici prostředí kompletního operačního systému. Chce výhodnou cenu a nechce se zapojovat do nastavování a údržby služeb sám ani dodatečnou platbou za správce.

Ideální webhostingová platforma vzniká sloučením právě kladných vlastností sdíleného a vyhrazeného webhostingu. Definoval jsem následující kritéria, která by měla multiwebhostingová platforma pro zákazníka splňovat.

Mutliwebhosting

- neomezené množství domén
- cena řádově desítky, maximálně stovky korun za měsíc
- vyhrazené systémové prostředky
- vyhrazený webový server, plně konfigurovatelný
- správcovský přístup možný, ale nikoliv nutný k používání
- zanedbatelná údržba ze strany zákazníka
- integrace do webové administrace poskytované firmou, automatizované administrační procesy

Multiwebhosting tedy obrazně zákazníkovi poskytuje vlastní webový server, který lze konfigurovat nezávislé na dalších zákaznících. Tento webový server má vyhrazeny systémové prostředky jako je operační paměť, čas procesoru, místo na disku a další. Je možné jej konfigurovat z firemní webové administrace. Cena by měla začínat v oblasti sdíleného webhostingu.

Na straně firmy musí být taková platforma pak snadno implementovatelná a efektivní. Musí tedy vhodně zacházet se systémovými zdroji, není možné, aby si například každý multiwebhostingový účet natvrdo ukrojil přidělenou operační paměť jako v případě vyhrazeného webhostingu. Platforma musí být snadná na správu a aktualizaci. Není možné ručně spravovat stovky jednotlivých serverů. Administrační procesy pracující nad multiwebhostingem musí být dobře automatizovatelné tak jako na sdíleném webhostingu. Co je administrační proces je uvedeno v kapitole 3.1.1.

3.1.1 Administrační procesy

V textu jsem již několikrát zmínil pojem administrační proces a jeho automatizace. Administrační procesy jsou akce, které jsou prováděny nad webhostingovým řešením za účelem vytvoření, zrušení a modifikace poskytovaných služeb. Administrační proces tedy může vytvářet nový nebo rušit stávající webhostingový účet. Pokud si zákazník ve firemní webové administraci změní nastavení svého webhostingu, jiný administrační proces se postará o odpovídající změnu konfigurace webhostingu.

Aby byly tyto procesy dobře implementovatelné, a aby byla možná jejich automatizace, tedy provedení bez zásahu lidské obsluhy ze strany firmy, je nutné dodržet určité požadavky na strukturu webhostingu. Prostředí zákaznického webhostingu musí být dostupné z místa, odkud mají být prováděny zásahy do jeho struktury a konfigurace. Administrační procesy by měly být vykonávány v hostitelském prostředí webhostingu, neměly by spoléhat na interakci s vnitřním prostředím zákaznického webhostingu. Důvod je jednoduchý, v prostředí zákaznického webhostingu může snadno dojít k chybě nebo změně ze strany zákazníka a administrační procesy by přestaly fungovat. Do prostředí hostitele není možné z vnitřního prostředí zákaznického webhostingu zasáhnout, tudíž není možné znemožnit funkci administračních procesů. Ty pak mohou mít ošetřeny i chybové stavy, kdy například zákazník poškodní své vnitřní prostředí webhostingu.

Nejčastěji je automatizace administračních procesů řešena na hostiteli webhostingu sadou různých skriptů. Ty se starají o založení struktur nových účtů a o přepis nebo doplnění konfigurace existujících webhostingů. Tyto skripty mohou být vzdáleně volány nebo periodicky spouštěny a nejčastěji na základě údajů z databáze provádějí potřebné kroky. Do databáze pak mohou být požadované akce zanášeny z uživatelsky příjemného prostředí firemní webové administrace.

3.2 Stanovení požadavků na virtualizační řešení

Z požadavků definovaných multiwebhostingovou platformou v kapitole 3.1 vyplývají kritéria, která musí splňovat virtualizační řešení, které bude použito pro implementaci této platformy. Než přikročím k popisu těchto kritérií, rád bych ještě rozebral otázku, zda je skutečně nutné pro implementaci multiwebhostingu zavádět virtualizaci.

Klasický sdílený webhosting sice umožňuje na jednom webovém serveru hostovat mnoho domén pro jednotlivé zákazníky, ale neposkytuje prostředky, jak pro tyto domény vyhradit systémové zdroje. Nelze tedy například pro 15 domén zákazníka číslo 9425 vyhradit 256 MB RAM. Dále není možné pro jednotlivé domény používat libovolná nastavení webového serveru. Mnoho z nastavení je sdílených pro všechny domény, což některé může omezovat. Za třetí, ale ne jako poslední je zde otázka bezpečnosti. Virtualizace vždy poskytne větší izolaci než oddělení na úrovni prostředků sdíleného webového serveru.

Vyhrazený webhostingový server ať již fyzický nebo virtuální zase nevyhovuje svou složitostí a cenou. V případě implementace webového rozhraní firmy pro použití s vyhrazeným webhostingem se tak sice pro zákazníka odboural složitá konfigurace a nároky na znalosti softwarového prostředí serveru, ale ze strany firmy je to stále další server, který si trvale alokuje cenné systémové prostředky, spotřebovává energii, a o který se musí někdo individuálně starat. Nemluvě o tom, že vyhrazený webhostingový server je poměrně dost izolovaný a automatizace administračních procesů nad ním není tak snadná.

Použití virtualizace pro implementaci multiwebhostingu dle stanovených požadavků je tedy nevyhnutelné. Není ale možné použít virtualizaci takovým způsobem, jakým je využívána pro vyhrazený webhosting.

Jak popíšu dále, díky speciálním požadavkům na virtualizační technologi nerozhoduje pouze její syrový výkon, který lze snadno změřit, ale také poskytovaná funkcionalita daného řešení. Zaměřím se především na omezování systémových zdrojů a strukturu hosta, která umožní snadnou automatizaci administračních procesů.

3.2.1 Omezování systémových zdrojů

Podstatným požadavkem na virtualizační řešení jsou možnosti omezování systémových zdrojů pro virtualizované hosty. Tím je možné nabídnout různé cenové tarify s různými výkonovými parametry. Zákazník, kterému pak nestačí 256 MB RAM pro provoz jeho domén si může připlatit za výkonnější řešení. Omezování systémových zdrojů také přispívá k zabezpečení služby. Jeden nebo několik zákazníků pak nemohou ať již cíleně nebo nevědomky zablokovat službu pro všechny ostatní.

Systémové zdroje je možné na platformě GNU/Linuxu omezit i nástroji třetí strany nebo vlastní realizací více či méně složitých mechanismů. Já zde ale budu při vyhodnocování možností

omezování zdrojů virtualizačních řešení brát v úvahu jen možnosti, které přinášejí přímo nástroje konkrétního virtualizačního produktu.

Procesorový čas

- Možnost nastavit počet fyzických procesorů pro hosta.
    - Hrubé odstupňování výkonu. Umožňuje zákazníkům poskytnout i dražší tarify s více procesory, což přispívá k navýšení výkonu, ale je také opticky příjemné.
- Možnost nastavit přidělení času procesoru, nejlépe v %.
    - Jemné odstupňování výkonu. Je možno přesně určit, kolik z času procesoru bude zákazníkovi přiděleno.
- Možnost nastavit prioritu pro jednotlivé hosty.
    - Někteří zákaznící mohou mít nastavenu vyšší prioritu. To je výhodné například při nabízení tarifů na zkoušku. Soupeří-li pak o čas procesoru řádný zákazník se zkušebním tarifem, je čas přednostně přidělen řádnému zákazníkovi.

Diskový prostor

- Možnost nastavit diskovou kvótu v bajtech.
    - Umožňuje pro jednotlivé cenové tarify nabídnout různé místo na disku. Zároveň disková kvóta chrání disk hostitele před jeho přeplněním.
- Možnost omezit počet použitých inodů.
    - Vhodné pro odstupňování cenových tarifů. Dražší tarify mohou povolovat uložení většího množství souborů a adresářů.
- Možnost nastavení I/O priority pro jednotlivé hosty.
    - Obdobně jako u procesorové priority. Je výhodné mít možnost zvýhodnit určitou skupinu zákazníku před jinou i v oblasti diskových operací.

Operační paměť

- Možnost nastavit maximální paměť v bajtech.
    - Vhodné pro odstupňování cenových tarifů. Dražší tarif s větší pamětí bude schopen zvládnout běh více domén s náročnějšími webovými aplikacemi.
- Alokace pouze aktuálně potřebné paměti a uvolňování cache.
    - Jednotliví virtualizovaní hosté musí alokovat pouze tolik paměti, kolik zrovna potřebují pro běh procesů. Jakmile proces doběhne, paměť je vrácena hostiteli. Není tedy možné, aby maximální paměť, kterou může host využít, byla neustále natvrdo odkrojena z dostupné paměti hostitele. Počítá se tedy s tím, že na hostiteli

poběží velké množství hostů, převyšující součtem maximální přípustné obsazené paměti dostupnou paměť fyzickou.

- GNU/Linuxové operační systémy standardně využívají celou dostupnou paměť pro různé cache a buffery, ne pouze pro aktuálně vykonávané procesy. Virtualizační řešení tedy musí zajistit uvolnění této cachované paměti v případě, že je paměť vyžadována pro běh dalších procesů jiného hosta, ale celá dostupná paměť hostitele je již obsazena.

Procesy

- Možnost omezit počet procesů.
  - Opět prostředek pro rozlišení různých cenových tarifů služby. Zároveň chrání před nekontrolovaným rozmnožením procesů hosta.

Síť

- Možnost omezit počet socketů.
  - Umožňuje nastavit pro různé tarify různé hodnoty. Také chrání před nekontrolovaným množstvím navázaných spojení.

3.2.2 Struktura multiwebhostingového hosta

Kromě požadavků na propracovanou limitaci systémových zdrojů přináší požadavky multiwebhostingové platformy také specifická kritéria v oblasti samotné struktury virtualizovaného hosta, který bude sloužit jako základní stavební jednotka multiwebhostingu.

- Podpora IP verze 4 a 6.
  - Síťová konektivita je důležitá už z principu určení. Do budoucna je také dobré mít k dispozici platformu, která podporuje IP verze 6.
- Viditelnost a možnost manipulace s procesy hostů.
  - Je dobré mít na hostiteli přístupné procesy všech virtualizovaných hostů. Lze toho využívat pro různé statistické údaje, například počítání procesů webového serveru Apache z jednoho místa pro všechny hosty. Tyto procesy je dobré mít možnost také ukončovat například v případě nějakého problému v hostovi.
- Instalace hosta v adresáři na souborovém systému hostitele.
  - Nezbytné pro zajištění automatizace administračních procesů nad hosty. Z hostitele pak může probíhat manipulace s daty v hostech. Například přidání nebo modifikace konfigurace webového serveru v hostovi. Rovněž zálohování se tímto velice usnadňuje.
- Možnost připojení různých adresářů z hostitele do prostředí hosta.

- Je možné pak například databázový server v hostovi umístit na rychlejší diskové pole, které není ovlivňováno dalšími nedatabázovými procesy.

- Možnost použít společnou šablonu operačního systému pro všechny hosty a odělení pouze individuální konfigurace a zákaznických dat
  - Je velice neefektivní, aby každý multiwebhostingový host představoval vyhrazený operační systém. Firma pak musí spravovat stovky dílčích serverů a takové řešení hraničí s vyhrazeným webhostingem. Místo toho je lepší, aby pro hosty existovala jedna společná šablona operačního systému a byla oddělena pouze zákaznická data a konfigurace. Zákazník má pak k dispozici výhody definované multiwebhostingem a ze strany firmy je spravována pouze jedna šablona operačního systému multiwebhostingových hostů.

3.3 Seznámení se s virtualizačními řešeními

V předchozích kapitolách jsem definoval požadavky, které multiwebhostingová platforma vznáší na použité virtualizační řešení. Aby bylo možno vybrat nejvhodnější řešení, musel jsem důkladněji prozkoumkat funkčnost vybraných virtualizačních řešení. Pro výběr nejvhodnějšího řešení nerozhoduje pouze výkon, ale také další kladené požadavky.

V tabulce 3.1 je uveden přehled vybraných virtualizačních řešení k prozkoumání a technika virtualizace, která u nich byla použita, jelikož například řešení Xen podporuje více typů virtualizace. Díky vhodnému výběru je zde zastoupena každá technika virtualizace, kromě výkonnostně velice slabé emulace. Rozhodující však nebude syrový výkon.

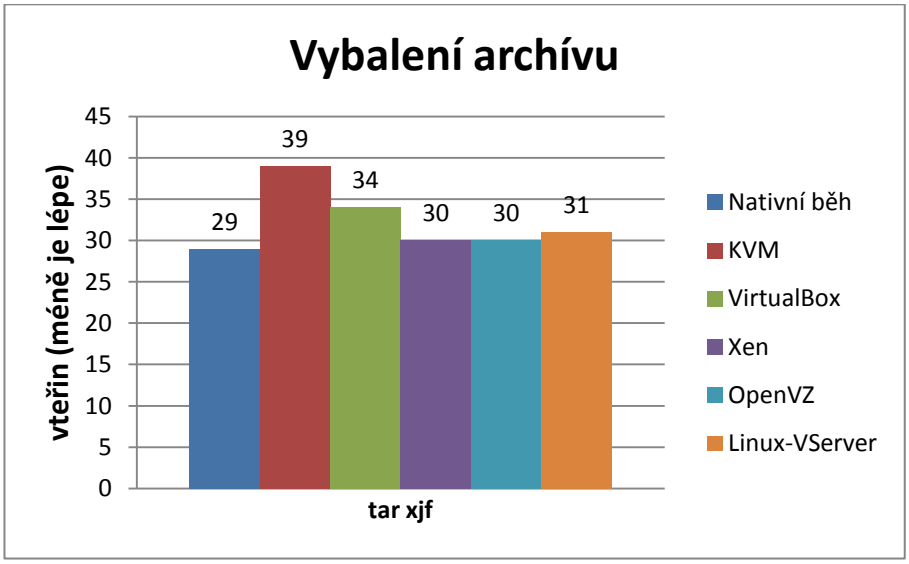
Řešení	Technika virtualizace	Verze
KVM	Hardwarově asistovaná plná	Jádro 3.2.11
Xen	Paravirtualizace	Xen 4.1.1-r2
VirtualBox	Hardwarově asistovaná plná	4.1.12-7724
OpenVZ	Na úrovni OS	Jádro 042stab053.5
Linux-Vserver	Na úrovni OS	Vserver 2.3.0.36.32

Tabulka 3.1 Techniky virtualizace jednotlivých řešení

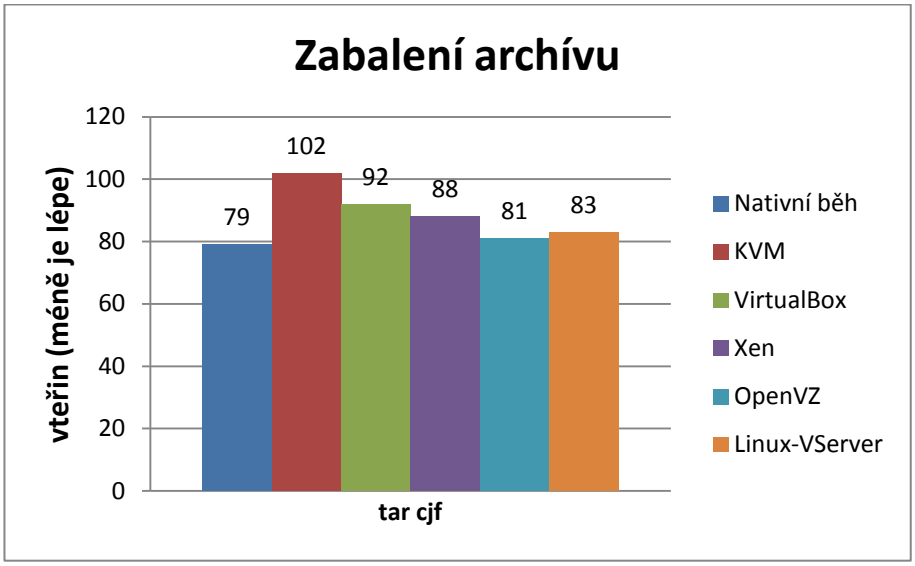
3.3.1 Výkonové testy

Všechny výkonové testy jsem prováděl na počítači třídy PC s operačním systémem Gentoo GNU/Linux. Počítač obsahoval čtyřjádrový procesor AMD, 8 GB operační paměti, gigabitovou síťovou kartu a dva pevné disky o 7200 otáčkách zapojené do diskového pole RAID typu 1. Pro všechny virtualizační řešení jsem se snažil zachovat stejné podmínky, tedy stejnou velikost přidělené opračí paměti a čas procesoru. Výsledky jsou uváděny jako průměry ze tří opakovaných měření. Verze použitých virtualizačních řešení jsou uvedeny v tabulce 3.1.

Prvním provedeným testem je rozbalení a zabalení archívu. Tento test je náročný na procesor a také na I/O operace zvláště tehdy, obsahují-li data archívu velké množství souborů, které je nutné číst a zapisovat. Rozbalovaloval a zabaloval jsem archív s Linuxovým jádrem komprimovaný *bzipem2* pomocí standardního příkazu *tar*. Delší čas se dá odhadovat především u řešení plné virtualizace. Výsledky jsou na obrázcích 3.1 a 3.2.



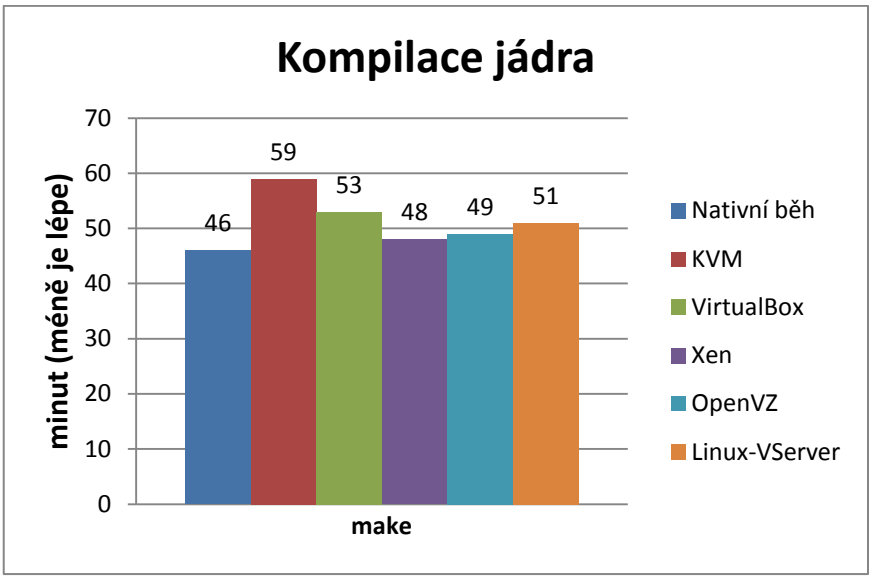
Obrázek 3.1 Výkon virtualizačních řešení při rozbalení archívu.



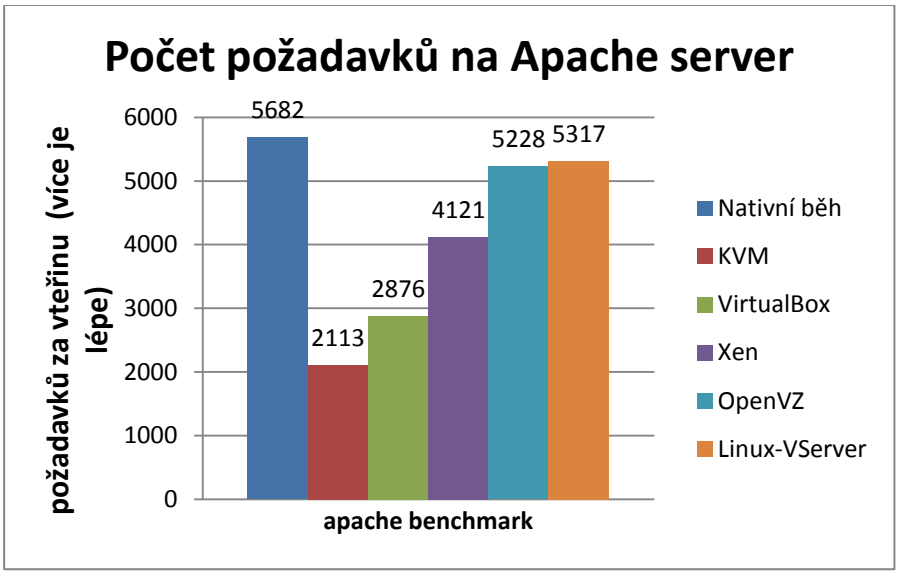
Obrázek 3.2 Výkon virtualizačních řešení při zabalení archívu.

Druhý test se týká kompilace jádra. Při této operaci je vytížen především procesor. Nedochází zde ke vzniku žádných privilegovaných operací, které by bylo nutné virtualizačními řešeními

odchytnout a ošetřit, rozdíl ve výkonu se dá tedy předpokládat minimální. Výsledky jsou na obrázku 3.3.



Obrázek 3.3 Výkon virtualizačních řešení při kompilaci jádra.

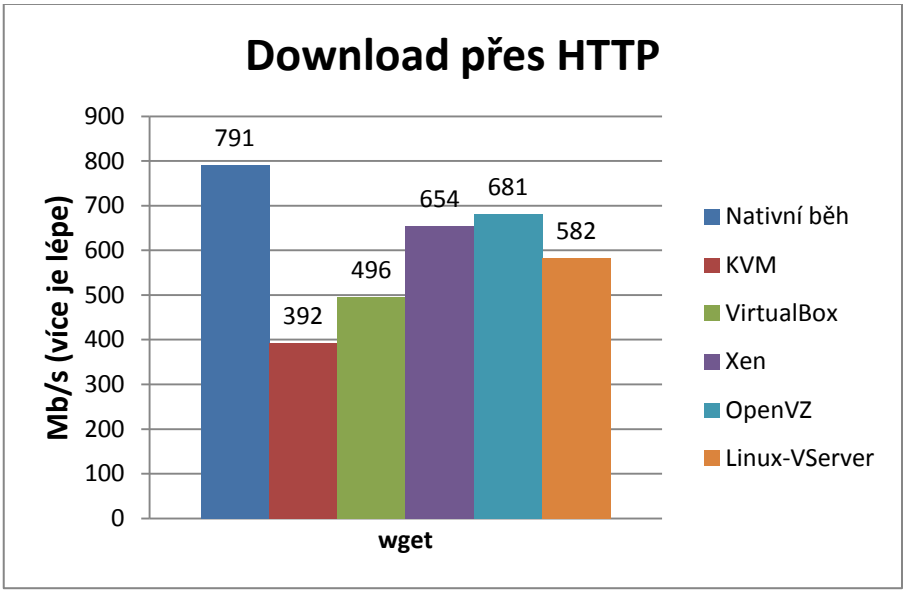


Obrázek 3.4 Výkon virtualizačních řešení při Aapache benchmarku.

Jelikož vybírané virtualizační řešení bude sloužit především pro aplikaci webového serveru, rozhodl jsem se také zařadit test nástrojem Apache Benchmark [17], který měří propustnost webového serveru Apache. Rozdíl ve výkonu je při tomto testu dosti podstatný. Řešení plné virtualizace KVM a VirtualBox se oproti nativnímu systému a dalším technikám virtualizace znatelně propadávají.

Nejblíže nativnímu systému pak mají řešení využívající techniku virtualizace na úrovni operačního systému – OpenVZ a Linux-VServer následované paravirtualizací - Xenem. Výsledky jsou na obrázku 3.4.

Posledním testem je měření propustnosti sítě na jednotlivých virtualizačních řešeních. Stahování velkého souboru probíhalo na gigabitové síti. Nejblíže téměř gigabitovému výkonu nativního systému pak mají opět řešení virtualizace na úrovní operačního systému – OpenVZ a Linux-VServer. Výsledky jsou na obrázku 3.5.



Obrázek 3.5 Výkon virtualizačních řešení v propustnosti sítě při stahování.

Obecně lze říci, že všechna vybraná virtualizační řešení jsou pro dnešní užití výkonově dostatečná, byť řešení využívající plné virtualizace s hardwarově asistovaným rozšířením zaostávají. To je daní za poskytované plné virtuální prostředí a možnost bez úprav použít libovolný operační systém.

Pro multiwebhostingovou platformu se tak z hlediska poskytovaného výkonu jeví jako nejlepší řešení virtualizace na úrovni operačního systému, kterou poskytuje OpenVZ a Linux-VServer nebo paravirtualizace zastoupená Xenem.

3.3.2 Funkční testy

Kromě změření výkonu jednotlivých virtualizačních řešení navzájem a vzhledem k nativnímu systému jsem se musel také dobře seznámit s poskytovanou funkčností a tzv. „aplikační filosofií" těchto virtualizačních řešení. Zaměřil jsem se především na požadavky formulované v kapitole 3.2. Postup zkoumání, který zahrnoval instalaci, konfiguraci, testování a studium dokumentace zde nebudu hlouběji rozebírat, podstatné závěry přináším v tabulce 3.2 a tabulce 3.3.

Sekce	CPU			Disk			RAM		Procesy	Síť
Řešení	Počet	%	Priorita	[B]	Inody	Priorita	[B]	Dynamická	Počet	Sockety
KVM	Ano	Ne	Ne	Ano	Ne	Ne	Ano	Ne	Ne	Ne
Xen	Ano	Ano	Ano	Ano	Ne	Ne	Ano	Ne	Ne	Ne
VirtualBox	Ano	Ano	Ne	Ano	Ne	Ne	Ano	Ne	Ne	Ne
OpenVZ	Ano	Ano	Ano	Ano	Ano	Ano	Ano	Ano	Ano	Ano
Linux-Vserver	Ano	Ano	Ano	Ano	Ano	Ne	Ano	Ano	Ano	Ano

Tabulka 3.2 Podpora limitace systémových zdrojů.

Vysvětlivky k tabulce 3.2:

Sekce *CPU* – Možnost omezovat počet fyzických CPU a procentuální přidělení času procesoru. Priorita se uplatňuje v případě, že o čas procesoru soupeří více hostů. Ten s nejvyšší prioritou je pak upřednostněn.

Sekce *Disk* – Možnost nastavit diskovou kvótu v bajtech pro dostupné místo a pro zabrané inody na disku. Priorita se pak uplatňuje v případě, že více hostů potřebuje přistupovat k disku.

Sekce *RAM* – Možnost nastavit maximální velikost obsazení paměti v bajtech. Sloupec *„Dynamická"* pak určuje, zda dané řešení podporuje alokaci paměti na vyžádání pouze pro běžící procesy nebo si natvrdo ukrojí nastavené využitelné maximum z dostupné paměti hostitele.

Sekce *Procesy* – Možnost omezit maximální počet spuštěných procesů v hostovi.

Sekce *Síť* - Možnost omezit počet použitých socketů.

Sekce	Procesy	Síť	Struktura hosta		
Řešení	Na hostiteli	IPv6	V adresáři	Mountování	Sdílená šablona
KVM	Ne	Ano	Ne	Ne	Ne
Xen	Ne	Ano	Ne	Ne	Ne
VirtualBox	Ne	Ano	Ne	Ne	Ne
OpenVZ	Ano	Ano	Ano	Ano	Ano
Linux-Vserver	Ano	Ano	Ano	Ano	Ano

Tabulka 3.3 Podpora speciálních vlastností.

Vysvětlivky k tabulce 3.3:

Sekce *Procesy* – Sloupec „*Na hostiteli“* udává, zda jsou procesy hostů dostupné hostiteli.

Sekce *Síť* – Podpora IP verze 6.

Sekce *Struktura hosta* - Sloupec „*V adresáři“* specifikuje, zda řešení podporuje instalaci, uzamčení a provoz hosta v adresáři na souborovém systému hostitele. Sloupec „*Mountování“* pak určuje, lze-li do prostředí virtuálního hosta mountovat jiné adresáře hostitele. Sloupec *„Sdílená šablona“* určuje, zda lze řešení přizpůsobit tak, aby bylo pro všechny hosty možné použít společnou šablonu operačního systému a oddělit pouze zákaznickou konfiguraci a data.

3.4 Volba virtualizačního řešení

Kromě poskytované funkčnosti, vhodné struktury a dostatečného výkonu jsou na virtualizační řešení kladeny ještě další nároky. Řešení musí být otevřené, aktivně dlouhodobě vyvíjené, stabilní a s dobrou dokumentační základnou. Do testování jsem zařadil pouze řešení, které tyto požadavky splňují.

Z výkonového hlediska jsou dle kapitoly 3.3.1 vyhovující řešení OpenVZ, Linux-VServer a Xen. Je však zapotřebí vzít v úvahu specifické požadavky definované v kapitole 3.2. Pokud tyto požadavky zkonzultujeme se zjištěnými výsledky v kapitole 3.3.2, vidíme, že jediná vyhovující řešení jsou OpenVZ a Linux-VServer využívající techniku virtualizace na úrovni operačního systému.

3.4.1 Paravirtualizace versus virtualizace na úrovni operačního systému

Paravirtualizace představovaná produktem Xen sice poskytuje dostatečný výkon, ale nesplňuje specifické požadavky na funkčnost virtualizačního řešení.

- Xen neumožňuje přidělování pouze aktuálně potřebné paměti pro běh procesů hostů. Každý host si vždy ukrojí maximální nastavenou dostupnou paměť. Není tedy možné spustit na hostiteli velké množství hostů, kteří si budou alokovat paměť hostitele jen dle aktuální potřeby.
- Xen nepodporuje instalaci hosta do adresáře na souborovém systému hostitele. Bylo by tedy nutné používat diskové obrazy nebo samostatné diskové oddíly LVM [18]. To komplikuje implementaci a znesnadňuje automatizaci administračních procesů nad jednotlivými hosty. Z hostitele není možné se přímým způsobem dostat k datům hosta.
- Xen nepodporuje připojování dalších míst z hostitele do hosta. Nelze tak například pro soubory databáze použít rychlejšího samostatného diskového pole.

- Xen neumožňuje použít sdílenou šablonu pro operační systém hosta a odělit pouze zákaznickou konfiguraci a data. Každý multiwebhosting by tedy představoval kompletní oddělený operační systém, což není vhodné pro snadnou a rychlou aktualizaci a údržbu.
- Dále pak Xen nepodporuje tak široké spektrum omezování systémových zdrojů hostů přímo na úrovni virtualizačního řešení. Chybí zde omezení počtu inodů a nastavení I/O priority. Nelze limitovat počet socketů a procesů.

3.4.2 OpenVZ versus Linux-VServer

Rozhodl jsem se tedy použít techniku virtualizace na úrovni operačního systému. Tato technika virtualizace je zde zastoupena ve dvou produktech. Oba produkty splňují výkonové i funkční testy provedené v kapitolách 3.3.1 a 3.3.2.

Rozhodl jsem se firmě doporučit virtualizační řešení OpenVZ z následujících důvodů:

- **Jádro**
  - Linuxové jádro použité v OpenVZ je založeno na Red Hat Enterprise Linuxu verze 6. Je pravdielně aktualizováno co do bezpečnostních záplat i do přidávání nových komponent. Lze tedy usuzovat, že jádro bude moderní a zároveň velmi stabilní.
- **Komerční podpora**
  - OpenVZ je podporováno komerční společností Parallels, která jej využívá ve svém vlastním proprietárním řešení Virtuozzo Containers a uvolňuje do něj některé pokročilé funkce. Přes to zůstává OpenVZ plně otevřeným řešením.
- **Snažší ovládání a použití**
  - Možná se jedná o subjektivní dojem, ale s OpenVZ se mi lépe pracovalo co do ovládání poskytovaných nástrojů, konfigurace i samotné filosofie aplikace.

3.4.3 Poznámky k terminologii OpenVZ

OpenVZ je virtualizační řešení, které využívá techniku virtualizace na úrovni operačního systému. Neposkytuje tedy kompletní virtuální prostředí, ale pouze prostor pro izolaci procesů hostovaného operačního systému a rozhraní pro přidělování systémových zdrojů hostitele. Více k této technice virtualizace jsem popsal v kapitolách 2.1.4 a 2.2.3. Výkon a podporované funkce jsou probrány v kapitolách 3.3.1 a 3.3.2.

OpenVZ přináší do světa virtualizace svou vlastní terminologii:

- **Kontejnerová virtualizace** – označení pro virtualizaci na úrovni operačního systému.

- **Hardwarový uzel** – označení pro hostitele.
    - Používá se zkratka *HN* - Hardware Node. Někdy také *CT0* – ConTainer 0, což označuje kontejner s nejnižším možným ID.
- **Kontejner** – označení pro hosta.
    - Používá se zkratka *CT* – ConTainer nebo *VE* – Virtual Environment. Za ním pak zpravidla pokračuje číslo kontejneru v rozsahu 0 až 65535. CT102 pak tvoří tzv. CTID, tedy ConTainer Identification. CT0 je vyhrazeno pro hardwarový uzel.
- **CTID** – identifikační číslo kontejneru, pod kterým vystupuje na hostiteli.
    - Je tvořeno zkratkou *CT* a číslem 0 az 65535, například CT102. CT0 je vyhrazeno pro hardwarový uzel HN a je doporučeno používat čísla od 101 výše.

4 Příprava prostředí virtuálního stroje

V této kapitole popíšu přípravu šablony operačního systému, která bude použita jako základ pro virtualizované hosty - kontejnery a její přizpůsobení k provozu na multiwebhostingové platformě. Jako operační systém pro hostitele i hosty jsem zvolil Linuxovou distribuci Gentoo GNU/Linux [19], jelikož ve stávající serverové farmě firmy zadavatele se používá výhradně tento operační systém. Příprava kontejneru počítá s funkčním virtualizačním prostředím OpenVZ na hostiteli. Více k instalaci hostitelského prostředí je v kapitole 7.

Nejprve je potřeba vytvořit základní OpenVZ šablonu založenou na Gentoo GNU/Linuxu. Poté se její softwarové prostředí upraví tak, aby odpovídalo poskytovaným službám multiwebhostingu. Jelikož není žádoucí, aby každý zákazník měl kompletní prostředí operačního systému, což by vedlo k velkému množství spravovaných virtuálních serverů, je následně nutné upravit šablonu tak, aby bylo možné používat pouze jednu kopii šablony pro všechny spouštěné kontejnery s individuálními zákaznickými daty a konfiguracemi webserverů.

4.1 Šablona pro operační systém kontejnerů

Virtualizační řešení OpenVZ nepoužívá pro vytvoření hostovaných operačních systémů standardní instalaci tak, jak ji známe například skrze instalační CD/DVD. Místo toho využívá tzv. šablony, což je archív se zabalenou souborovou strukturou GNU/Linuxové distribuce, která je vhodně zkonfigurována pro provoz pod OpenVZ.

Distribuce Gentoo GNU/Linux používá obdobný systém instalace i na fyzický počítač. Soubory této distribuce jsou šířeny jako tzv. stage3 archív, který se nakopíruje na disk počítače, upraví se jeho konfigurace, nainstaluje se do něj Linuxové jádro, na disk se nainstaluje zavaděč a následně je možné operační systém nabootovat. Toto usnadňuje vytvoření vlastní šablony pro OpenVZ. Stačí si z oficiálních stránek Gentoo GNU/Linuxu stáhnout aktuální stage3 archív, rozbalit jej, upravit a doinstalovat software.

Na stránkách projektu OpenVZ lze nalézt oficiální návod pro vytvoření Gentoo GNU/Linuxové šablony [20], nicméně ten není zrovna aktuální a po nedávném přechodu Gentoo na Baselayout 2 a init systém OpenRC [21] neprodukuje funkční OpenVZ šablonu. Pro přípravu funkční šablony jsem tedy musel přijít s vlastním postupem, který se přidržuje oficiálního návodu, ale je upraven tak, aby byl použitelný pro aktuální Gentoo GNU/Linux stage3.

4.1.1 Vytvoření základní šablony

Stáhneme aktuální stage3 archív z Gentoo mirroru. Vytvoříme instalační kontejner s ID 777 a rozbalíme do něj stažený stage3 archív:

```
(hostitel)# mkdir /vz/private/777
(hostitel)# tar -xjf stage3-amd64-latest.tar.bz2 -C
/vz/private/777
```

Odvodíme konfiguraci kontejneru 777 z „konfiguračního vzoru“ ve-vswap-256m.conf-sample. Ten obsahuje základní definice přidělených zdrojů.

```
(hostitel)# vzctl set 777 --applyconfig vswap-256m –save
```

Přidáme název šablony – „gentoo“ bude odpovídat archívu šablony „gentoo.tar.gz“.

```
(hostitel)# vi /etc/vz/conf/777.conf
   OSTEMPLATE="gentoo"
```

Vytvoříme z konfigurace instalačního kontejneru 777 obecný konfigurační soubor pro Gentoo kontejnery.

```
(hostitel)# cp /etc/vz/conf/777.conf /etc/vz/conf/ve-gentoo.conf-
sample
```

Nyní začneme s úpravou vlastního systému pro kontejner. Kořenový oddíl je připojován hostitelem a nikoliv hostem, takže soubor */etc/mtab* nasměrujeme na */proc/mounts*, kde se objeví korektní záznamy.

```
(hostitel)# rm -f /vz/private/777/etc/mtab
(hostitel)# ln -s /proc/mounts /vz/private/777/etc/mtab
```

Přepneme se *chrootem* do prostředí kontejnerového systému.

```
(chroot)# chroot /vz/private/777 /bin/bash
```

Vytvoříme chybějící adresáře pro strom portage.

```
(chroot)# mkdir /opt /usr/portage /etc/portage
```

Upravíme soubor */etc/fstab*. Uvnitř kontejneru je potřeba pouze */proc*. Ostatní mountování provádí hostitel.

```
(chroot)# echo "proc /proc proc defaults 0 0" > /etc/fstab
```

Upravíme */etc/inittab*. Vykomentujeme všechny řádky pro *tty*. Kontejner nemá fyzické konzole k dispozici.

```
(chroot)# vi /etc/inittab
   #c1:12345:respawn:/sbin/agetty 38400 tty1 linux
   #c2:2345:respawn:/sbin/agetty 38400 tty2 linux
   #c3:2345:respawn:/sbin/agetty 38400 tty3 linux
   #c4:2345:respawn:/sbin/agetty 38400 tty4 linux
   #c5:2345:respawn:/sbin/agetty 38400 tty5 linux
   #c6:2345:respawn:/sbin/agetty 38400 tty6 linux
```

Vygenerujeme patřičné locales, zde používám anglické UTF-8.

```
(chroot)# vi /etc/locale.gen
   en_US ISO-8859-1
   en_US.UTF-8 UTF-8

(chroot)# locale-gen
```

Nastavíme časové pásmo na Českou republiku.

```
(chroot)# cp /usr/share/zoneinfo/Europe/Prague /etc/localtime
(chroot)# echo "Europe/Prague" > /etc/timezone
```

Všechny systémové locales nastavíme na anglické UTF-8.

```
(chroot)# vi /etc/env.d/02locale
   LANG="en_US.UTF-8"
   LC_ALL="en_US.UTF-8"
```

Upravíme soubor */etc/rc.conf*. Je nutné nastavit *rc_sys* na prázdnou hodnotu, jinak nebude možno šablonu použít ve sdíleném režimu kontejnerů dle kapitoly 4.3.

```
(chroot)# vi /etc/rc.conf
   rc_sys=""
```

Upravíme soubor */etc/make.conf*, který specifikuje hlavní proměnné prostředí používáné v Gentoo GNU/Linuxu. Zde je uvedena obecná kostra souboru *make.conf*.

```
(chroot)# vi /etc/make.conf
   CFLAGS="-O2 -pipe -fomit-frame-pointer -march=native"
   CXXFLAGS="${CFLAGS}"
   CHOST="x86_64-pc-linux-gnu"
   MAKEOPTS="-j2"
   FEATURES="parallel-fetch"
   LINGUAS="cs en"
   GENTOO_MIRRORS="http://gentoo.mirror.dkm.cz/pub/gentoo/ "
   SYNC="rsync://rsync.europe.gentoo.org/gentoo-portage"
   CONFIG_PROTECT="/sbin/rc"
   PHP_TARGETS="php5-3 php5-2"
   APACHE2_MPMS="worker"
   USE="ipv6 mmx nls iproute2 sse sse2 ssl unicode -X -alsa -atm
        -bluetooth -gnome -gtk -kde kerberos -ldap -msql -mssql
        -oracle -oss -qt4 -qt3 -qt -radius -ruby -tcl -xinetd
        -sqlite -sqlite3"
```

Upravíme soubor */etc/pam.d/chpasswd*, aby bylo možno měnit hesla přes nástroje OpenVZ. Změníme řádek *password*.

```
(chroot)# vi /etc/pam.d/chpasswd
   password required pam_unix.so md5 shadow
```

Aktualizujeme prostředí šablony a opustíme *chroot* prostředí kontejneru.

```
(chroot)# env-update
(chroot)# source /etc/profile
(chroot)# exit
```

Instalační kontejner můžeme nyní pod OpenVZ hostitelem nastartovat a vstoupit do něj. Připojíme do kontejneru také strom portage pro provedení aktualizace a doinstalování softwarového prostředí kontejneru. Lze použít i vlastní strom portage uvnitř kontejneru (klasická instalace *emerge portage*), ale to my z důvodu úspory místa na disku dělat nebudeme a použijeme místo toho připojení portage stromu hostitele.

```
(hostitel)# cd
(hostitel)# vzctl start 777
(hostitel)# emerge --sync -q
(hostitel)# mount -n -t simfs /usr/portage \
/vz/root/777/usr/portage -o /usr/portage
(hostitel)# vzctl enter 777
```

Provedeme aktualizaci Gentoo GNU/Linuxu v kontejneru.

```
(kontejner)# emerge --NuDa world
(kontejner)# etc-update
```

Nepotřebné init skripty pro kontejnerové prostředí vyřadíme ze spouštění při startu kontejneru.

```
(kontejner)# rc-update del keymaps boot
(kontejner)# rc-update del udev sysinit
(kontejner)# rc-update del netmount default
(kontejner)# rc-update del hwclock boot
(kontejner)# rc-update del modules boo
(kontejner)# rc-update del fsck boot
(kontejner)# rc-update del root boot
(kontejner)# rc-update del mtab boot
(kontejner)# rc-update del termencoding boot
(kontejner)# rc-update del swap boot
(kontejner)# rc-update del procfs boot
(kontejner)# rc-update del urandom boot
(kontejner)# rc-update del dmesg sysinit
(kontejner)# rc-update del udev-postmount default
```

Nainstalujeme požadované softwarové prostředky do kontejneru.

```
(kontejner)# emerge -av vim htop app-misc/mc iproute2 ...
```

Vystoupíme z konzole kontejneru zpět do hostitele.

```
(kontejner)# exit
```

Zastavíme instlační kontejner 777 a vytvoříme z něj archív s OpenVZ šablonu.

```
(kontejner)# vzctl stop 777
(kontejner)# cd /vz/private/777/
(kontejner)# tar --numeric-owner -czf \
/vz/template/cache/gentoo.tar.gz *
```

Tuto šablonu je nyní možné použít pro vytváření úplných nových kontejnerů založených na Gentoo GNU/Linuxu. Příklad vytvoření kontejneru s ID 101 je níže.

```
(hostitel)# vzctl create 101 --config gentoo
```

4.2 Softwarové prostředí šablony kontejneru

Softwarové prostředí šablony se odvíjí od jejího použití. Já připravuji šablonu pro mutliwebhosting a takové prostředí má své typické požadavky. Dle standardní webhostingové nabídky firmy bude multiwebhosting poskytovat tyto služby:

- Webový server Apache s podporou jazyka PHP
- Databázový server MySQL

Do kontejneru je tedy třeba doinstalovat software Apache s PHP a databázový server MySQL. Konfigurace služeb je odvozena dle požadavku zadavatele od stávajícího sdíleného webhostingu, přičemž klient bude mít možnost plné změny konfigurace webového serveru. Databázový server bude v nastavení dle cenových tarifů omezován co do současných připojení, možností využít InnoDB engine apod.

Přístup k datům bude řešen FTP serverem, který bude instalován na hostiteli. Pro mailový hosting bude využito stávajícího mailového systému firmy, do kontejneru tedy nebude žádný mailový subsystém instalován.

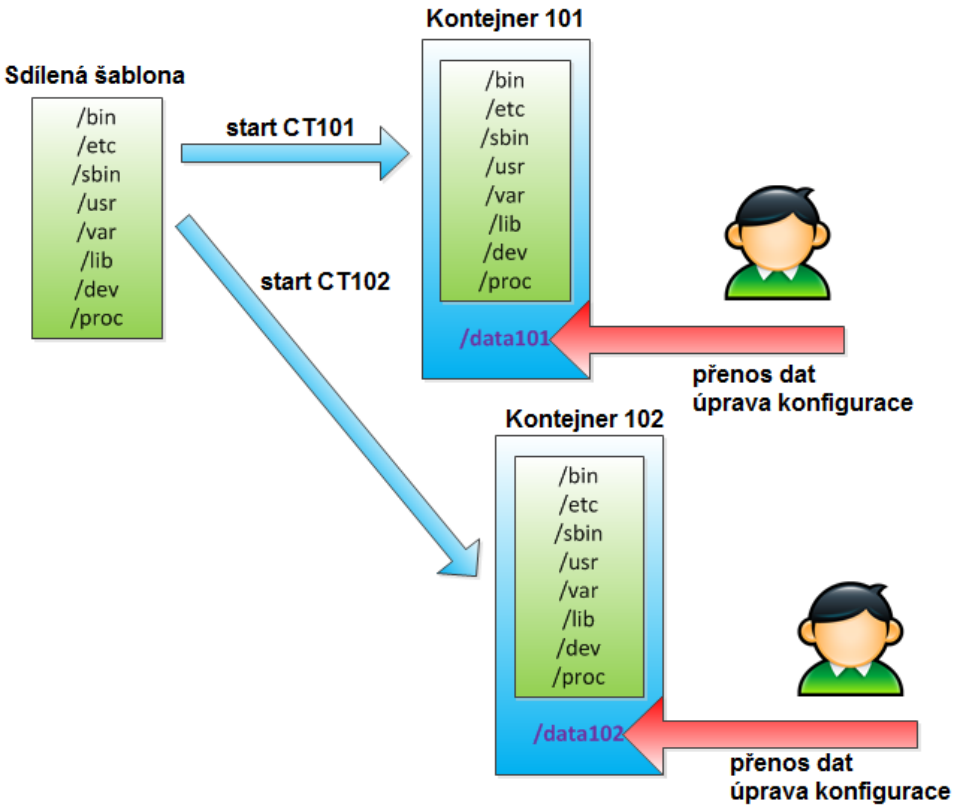
4.3 Sdílený režim kontejnerů

Jelikož je nutné zachovat snadnou správu zákaznických kontejnerů ze strany obsluhy firmy zadavatele, není možné pro každého zákazníka vytvořit plnohodnotný kontejner. To není nutné ani ze strany zákazníka, ten potřebuje pouze široce konfigurovatelný webový a databázový server, nikoliv kompletní prostředí virtuálního operačního systému umístěného v kontejneru.

Nabízí se tedy myšlenka, proč nedonutit virtualizační řešení k tomu, aby umožňovalo jednotlivé kontejnery spouštět ze společné šablony operačního systému, ale zároveň oddělit individuální konfiguraci služeb a zákaznická data. Pokud se toto povede, získáme tím následující výhody.

- Úspora místa na disku, operační systém pro kontejner jako celek je přítomen fyzicky pouze jednou.
- Snadná aktualizace, stačí provést aktualizaci společné šablony.
- Snadná globální konfigurace, stačí změnit nastavení šablony.

Princip jak dosáhnout ve virtualizačním řešení OpenVZ takového chování je vícenásobné spuštění sdílené šablony jako kontejneru, který však obsahuje také individuální nesdílenou oblast. Takový způsob provozu kontejnerů jsem nazval jako **sdílený režim kontejnerů**.



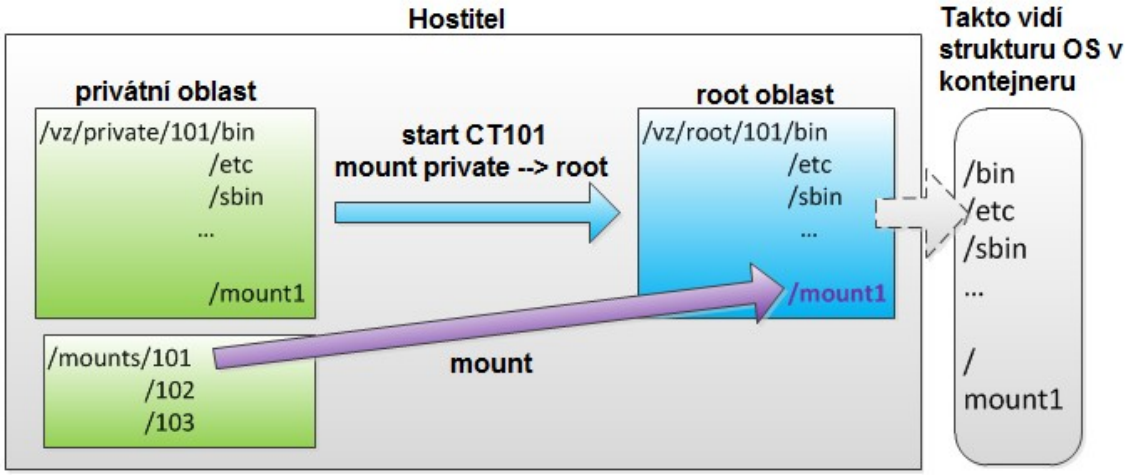
Obrázek 4.1 Princip sdílené šablony pro kontejnery.

Na obrázku 4.1 je zachycena idea sdíleného režimu kontejnerů. Na disku hostitele je přítomna pouze jedna sdílená šablona. Ta obsahuje soubory operačního systému (adresáře */bin*, */etc*, */lib*, atd.). Tuto šablonu spustíme vícekrát pro všechny zákazníky. Tím dosáhneme stejného softwarového prostředí všech spouštěných kontejnerů, ještě je však nutné zajistit, aby výsledný kontejner obsahoval i individuální oblast pro uložení konfigurací a dat (*/data/101*, */data102*), na které si kontejnery navzájem nevidí.

4.3.1 Úprava základní šablony na sdílenou

Základní šablonu pro multiwebhostingové kontejnery je potřeba upravit pro provoz ve sdíleném režimu kontejnerů dle idey z obrázku 4.1. Než popíšu provedené úpravy, je potřeba říct něco málo ke strategii, jakou OpenVZ spouští kontejnery a provádí připojování oblastí.

Fyzická data operačních systémů pro spuštění kontejnerů se nalézají v adresáři hostitele */vz/private* pod ID kontejneru. Po spuštění kontejneru je tato oblast připojena do adresáře */vz/root* pod stejným ID. Jakmile je kontejner připojen ve */vz/root* mohou se do této oblasti připojovat další adresáře z hostitele. Toto znázorňuje obrázek 4.2.



Obrázek 4.2 Připojení kontejneru během jeho startu.

Každý OpenVZ kontejner má tedy v základu vždy právě dvě oblasti:

- **/vz/private/${CTID} (privátní oblast)** – data operačního systému pro kontejner, po startu se připojují do root oblasti.
- **/vz/root/${CTID} (root oblast)** – sem se kontejner po startu připojí a je zde možné připojovat další oblasti z hostitele.

Privátní oblast tedy ve výchozím chování OpenVZ obsahuje data operačního systému kontejneru. Root oblast pak po nastartování obsahuje to, co kontejner po úplném spuštění operačního systému vidí jako svou konečnou kořenovou adresářovou strukturu, tedy připojený privátní prostor, na který mohou být připojeny další oblasti z hostitele. Na obrázku 4.2 je například na adresář */vz/root/101/mount1* kontejneru připojena jiná oblast hostitele */mounts/101*.

Při vícenásobném spouštění jedné šablony jsem však narazil na problém, že OpenVZ neumožňuje se zapnutými diskovými kvótami jednu privátní oblast připojit najednou vícekrát (při

takovém pokusu nedojde ke spuštění druhého a dalších kontejnerů s chybou, že privátní oblast je již jednou připojena). Je tedy nutné zachovat individuální privátní oblast pro každý kontejner. Musel jsem tedy přijít s řešením, které by toto obcházelo, jelikož oželet možnost limitovat diskový prostor a počet zabraných inodů kontejnerem na úrovni OpenVZ jsem považoval za nepřípustné a používat pro každou privátní oblast samostatnou úplnou šablonu operačního systému odporuje dříve stanoveným, požadavkům.

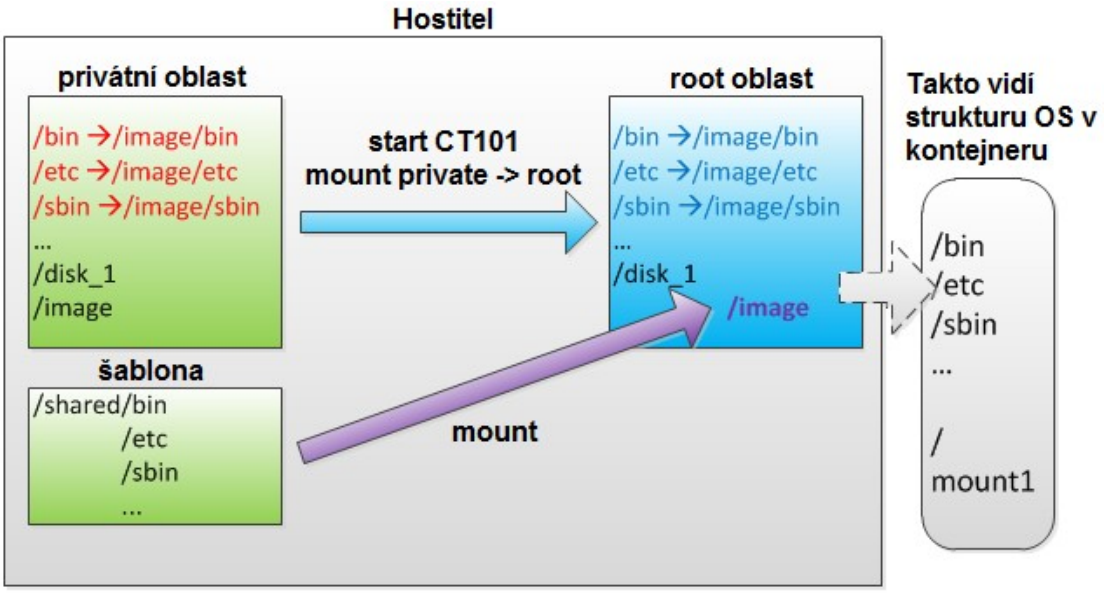
Napadlo mě tedy zachovat oddělenou privátní oblast pro každý kontejner a šablonu operačního systému připojit do root oblasti po nastartování kontejneru. Pro kontejner jsem připravil privátní oblast, která sestává ze symbolických odkazů, které svou strukturou odpovídají kořenové struktuře šablony operačního systému kontejneru. Tyto odkazy jsem pak nasměroval do prázdného adresáře */image* privátní oblasti, viz obrázek 4.3. Není nutné a dokonce ani vhodné vytvářet všechny adresáře privátní oblasti jako linkované. Pro dynamicky vytvářené struktury jako je */proc*, */sys* a */dev* stačí ponechat prázdný adresář a jádro se po startu postará o jejich korektní naplnění.

```
lrwxrwxrwx   1 root root    10 Jan  6 09:12 bin -> /image/bin
lrwxrwxrwx   1 root root    11 Jan  6 09:13 boot -> /image/boot
drwxr-xr-x+  5 root root 4.0K Dec 20 09:16 dev
drwxr-xr-x+  6 root root 4.0K Apr  2 15:25 disk_1
lrwxrwxrwx   1 root root    10 Jan  6 09:13 etc -> /image/etc
-rw-r--r--   1 root root     0 Jan  6 09:20 fastboot
lrwxrwxrwx   1 root root    11 Jan  6 09:13 home -> /image/home
drwxr-xr-x   2 root root 4.0K Jan  6 09:09 image
lrwxrwxrwx   1 root root    12 Jan  6 09:14 lib -> /image/lib64
lrwxrwxrwx   1 root root    12 Jan  6 09:14 lib32 -> /image/lib32
lrwxrwxrwx   1 root root    12 Jan  6 09:14 lib64 -> /image/lib64
drwxr-xr-x+  2 root root 4.0K Jan  6 09:10 mnt
lrwxrwxrwx   1 root root    10 Jan  6 09:14 opt -> /image/opt
drwxr-xr-x   2 root root 4.0K Jan  6 09:10 proc
lrwxrwxrwx   1 root root    11 Jan  6 09:14 root -> /image/root
lrwxrwxrwx   1 root root    11 Jan  6 09:28 sbin -> /image/sbin
drwxr-xr-x   2 root root 4.0K Jan  6 09:10 sys
lrwxrwxrwx   1 root root    11 Jan  6 09:46 tmp -> /tmp_vz/tmp
lrwxrwxrwx   1 root root    14 Jan  6 09:58 tmp_db -> /tmp_vz/tmp_db
drwxr-xr-x   2 root root 4.0K Jan  6 09:32 tmp_session
drwxr-xr-x   2 root root 4.0K Jan  6 09:32 tmp_vz
lrwxrwxrwx   1 root root    15 Jan  6 09:58 tmp_www -> /tmp_vz/tmp_www
lrwxrwxrwx   1 root root    10 Jan  6 09:14 usr -> /image/usr
lrwxrwxrwx   1 root root    10 Jan  6 10:29 var -> /image/var
```

Obrázek 4.3 Privátní oblast multiwebhostingového kontejneru.

Spousta nefunkčních symbolických odkazů vedoucích do adresáře */image* je správně. Do adresáře */image* po startu kontejneru provedu v root oblasti připojení struktury sdílené šablony operačního systému pro kontejner. To zajistí, že symbolické odkazy v root oblasti budou odkazovat na

skutečná data a root oblast se bude chovat jako korektní kontejner s operačním systémem. Viz obrázek 4.4.



Obrázek 4.4 Připojení sdílené šablony do root oblasti.

```
lrwxrwxrwx   1 root root   10 Jan  6 09:12 bin -> /image/bin
lrwxrwxrwx   1 root root   11 Jan  6 09:13 boot -> /image/boot
drwxr-xr-x+  5 root root 4.0K Dec 20 09:16 dev
drwxr-xr-x+  6 root root 4.0K Apr  2 15:25 disk_1
lrwxrwxrwx   1 root root   10 Jan  6 09:13 etc -> /image/etc
-rw-r--r--   1 root root    0 Jan  6 09:20 fastboot
lrwxrwxrwx   1 root root   11 Jan  6 09:13 home -> /image/home
drwxr-xr-x+ 21 root root 4.0K Feb 17 12:21 image
lrwxrwxrwx   1 root root   12 Jan  6 09:14 lib -> /image/lib64
lrwxrwxrwx   1 root root   12 Jan  6 09:14 lib32 -> /image/lib32
lrwxrwxrwx   1 root root   12 Jan  6 09:14 lib64 -> /image/lib64
drwxr-xr-x+  2 root root 4.0K Jan  6 09:10 mnt
lrwxrwxrwx   1 root root   10 Jan  6 09:14 opt -> /image/opt
dr-xr-xr-x  46 root root    0 Feb 23 09:14 proc
lrwxrwxrwx   1 root root   11 Jan  6 09:14 root -> /image/root
lrwxrwxrwx   1 root root   11 Jan  6 09:28 sbin -> /image/sbin
drwxr-xr-x   6 root root    0 Feb 23 09:14 sys
lrwxrwxrwx   1 root root   11 Jan  6 09:46 tmp -> /tmp_vz/tmp
lrwxrwxrwx   1 root root   14 Jan  6 09:58 tmp_db -> /tmp_vz/tmp_db
drwxr-xr-x   2 web  web    40 Feb 23 09:14 tmp_session
drwxr-xr-x   8 root root 4.0K Jan  6 09:34 tmp_vz
lrwxrwxrwx   1 root root   15 Jan  6 09:58 tmp_www -> /tmp_vz/tmp_www
lrwxrwxrwx   1 root root   10 Jan  6 09:14 usr -> /image/usr
lrwxrwxrwx   1 root root   10 Jan  6 10:29 var -> /image/var
```

Obrázek 4.5 Struktura / nastartovaného kontejneru.

Pokud takový kontejner spustíme a prohlédneme si jeho kořenový adresář, uvidíme strukturu jako na obrázku 4.5. V adresáři */image* je pak skutečně připojena sdílená šablona kontejnerů, která již fyzicky obsahuje strukturu operačního systému. Viz obrázek 4.6.

```
m9227 / # ls image/
bin   dev     etc       home  lib32  media  opt   root  sys  tmp_db       tmp_vz   usr
boot  disk_1  fastboot  lib   lib64  mnt    proc  sbin  tmp  tmp_session  tmp_www  var
```

Obrázek 4.6 Struktura /image nastartovaného kontejneru.

Vytvořením privátní struktury symbolických odkazů ukazujících do adresáře, kam bude po startu kontejneru připojena sdílená šablona operačního systému je tedy možné obejít nemožnost spouštět pod OpenVZ více kontejnerů z jedné společné privátní oblasti při aktivních diskových kvótách. OpenVZ si při startu kontejneru nestěžuje, že privátní oblast je již jednou připojena do root oblasti, ale přitom využíváme stále stejnou šablonu pro mnoho kontejnerů.

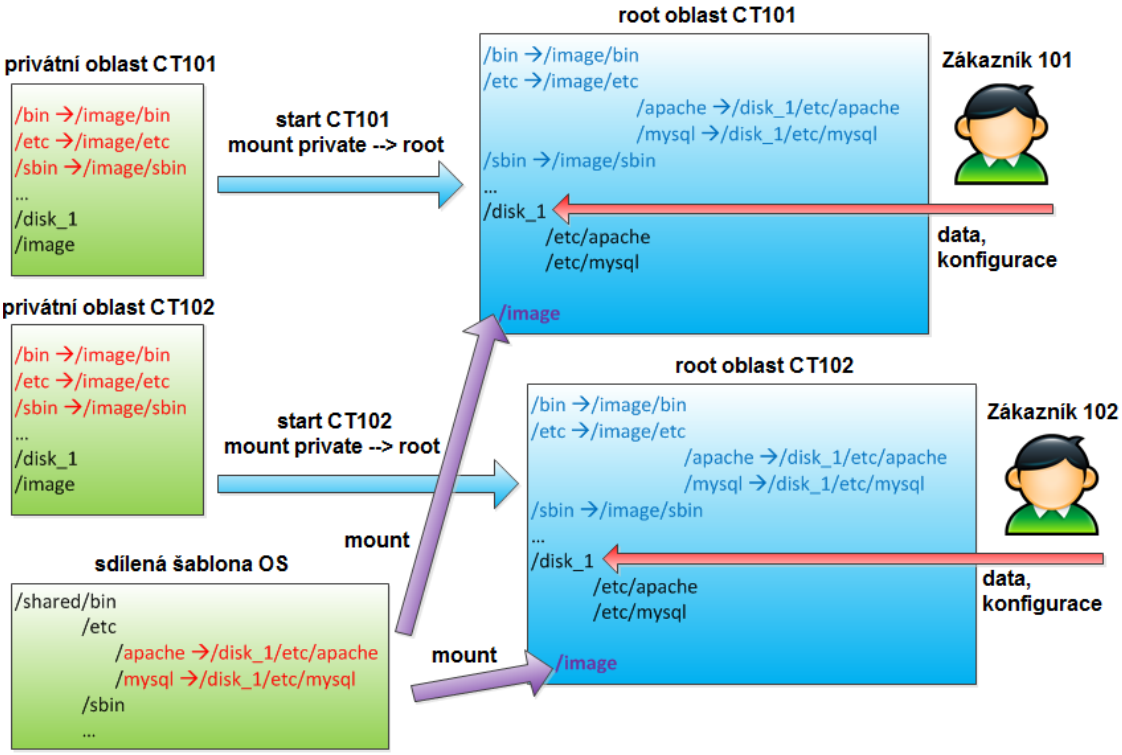
Díky takto navržené struktuře zůstávají navíc privátní oblasti kontejnerů na sobě nezávislé a je možné je použít pro ukládání zákaznických dat a konfigurací. Například adresář */disk_1* na obrázku 4.4 je součástí privátní oblasti, která je oddělena pro všechny kontejnery. Po startu kontejneru a jeho připojení na root oblast, je tento adresář k dispozici pro individuální využití běžícím kontejnerem.

Aby se aplikovaly konfigurační soubory umístěné v zákaznické datové oblasti a ne ty globální v adresáří */etc*, resp. */image/etc* sdílené šablony, je potřeba ve struktuře šablony provést nalinkování na zákaznickou konfiguraci. Nalinkování se pak týká konfiguračních souborů webového serveru Apache, databázového serveru MySQL a některých dalších systémových konfiguračních souborů, jako je třeba nastavení sítě, hostname, apod.

Na obrázku 4.7 je uveden příklad linkování */etc/my.cnf* pro konfiguraci MySQL. Adresář */etc* je linkován do adresáře šablony */image/etc* dle výše popsaného principu a soubor šablony */image/etc/mysql/my.cnf* je linkován do zákaznické oblasti *v /disk_1*.

```
m9227 mysql # ls -lah /etc
lrwxrwxrwx 1 root root 10 Jan  6 09:13 /etc -> /image/etc
m9227 mysql # ls -lah /image/etc/mysql/my.cnf
lrwxrwxrwx 1 root root 24 Jan 13 09:38 /image/etc/mysql/my.cnf -> /disk_1/etc/mysql/my.cnf
```

Obrázek 4.7 Linkovaná konfigurace pro MySQL.



Obrázek 4.8 Kompletní struktura sdíleného režimu kontejnerů.

Na obrázku 4.8 je znázorněna kompletní struktura pro sdílený režim kontejnerů. Každý z kontejnerů 101 a 102 má tedy svou vyhrazenou privátní oblast, která obsahuje několik symbolických odkazů na strukturu operačního systému a adresář */image* pro připojení sdílené šablony. Po startu kontejneru se připojí privátní oblast do root oblasti a také se provede připojení sdílené šablony operačního systému, která je společná pro všechny spouštěné kontejnery. Symbolické odkazy privátní oblasti se tak po připojení do root oblasti stanou aktivními. Privátní oblast obsahuje dále adresář */disk_1* s daty a individuálními konfiguracemi kontejneru. Na tyto individuální konfigurace jsou směrovány odkazy ze sdílené šablony pro konfigurace služeb, jež mají být individuálně nastavitelné (Apache, MySQL). Zákazník má tak tedy k dispozici kontejner, který poskytuje plně konfigurovatelný webový a databázový server, ale ze strany firmy je pro všechny kontejnery použita společná šablona operačního systému s instalovaným softwarem.

4.3.2 Doplnění konečné struktury pro zákaznický kontejner

S funkčním sdíleným režimem kontejnerů jsem se mohl začít věnovat návrhu kompletní konečné struktury pro multiwebhostingové kontejnery. V nejjednodušším možném případě si sice struktura vystačí s připojením sdílené šablony a vhodným prolinkováním, já jsem se však zajímal, jak tuto strukturu bezpečnostně a výkonnostně zdokonalit dále.

Z hlediska výkonnosti jsem se rozhodl umístit soubory databázových serverů MySQL na vyhrazené diskové pole. Tím jsem zajistil, že databázové operace nebudou rušeny dalšími I/O požadavky. Pro strukturu kontejneru se toho z hlediska organizace mnoho nezmění. Pouze po nastartování kontejneru se nebude připojovat jen oblast se sdílenou šablonou, ale také oblast s databázovými soubory. Tato oblast je pak umístěna na vhodné části hostitele, jako je třeba připojené samostatné diskové pole.

Více k diskové struktuře je řečeno v kapitole 7.

4.3.3 Souborové zabezpečení šablony a aplikace ACL

V zájmu dosažení maximální bezpečnosti jsem se rozhodl, že sdílená šablona operačního systému se bude po spuštění kontejneru připojovat pouze v režimu pro čtení. Zákazník nemá důvod do sdílené šablony cokoliv zapisovat nebo v ní cokoliv měnit. Došlo by k ovlivnění globální konfigurace nebo poškození sdílené šablony využívané všemi kontejnery. Pokud se tedy pokusíme do šablony z kontejneru něco zapsat, systém zahlásí souborový systém pouze pro čtení. Viz obrázek 4.9.

```
m9227 / # touch /image/ZAPISUJI
touch: cannot touch `/image/ZAPISUJI': Read-only file system
m9227 / # touch /image/etc/ZAPISUJI
touch: cannot touch `/image/etc/ZAPISUJI': Read-only file system
```

Obrázek 4.9 Pokus o zápis do adresáře se šablonou.

Zákaznická data jsou zapisovatelná, ale není na nich povoleno spouštění souborů (připojovací volba volba *noexec*) a suid (připojovací volba *nosuid*). Možnost využít těchto připojovacích voleb poskytuje připojení adresáře se zákaznickými daty sama na sebe po startu kontejneru.

Jelikož ze struktury šablony i kontejneru je možné zjistit další údaje o tom, jak je multiwebhostingová platforma navržena (například průzkum konkurence), přistoupil jsem také k dodatečné aplikaci souborových access listů – ACL. Standardními Linuxovými právy není možné dostatečně dobře a snadno omezit pro uživatele, pod kterým běží webový server, přístup do oblastí, které nejsou vyhrazeny k užití zákazníkem. Pomocí tzv. webového shellu, např. r57 [22] pak má zákazník možnost zkoumat přes webové stránky svého webhostingu souborový systém kontejneru a spouštět dokonce i příkazy.

Souborové ACL umožňují na podporovaných souborových systémech nastavovat pravidla pro konkrétní uživatele a skupiny. Standardní systém Linuxových práv dle POSIX umožňuje definovat pouze práva pro vlastníka, skupinu a všechny ostatní.

Webový server je spouštěn pod uživatelem web. Při návrhu souborových ACL jsem postupoval tak, že jsem nejprve pro uživatele web vše zakázal a postupně jsem povoloval jen to, k čemu by měl mít přístup, typicky konfigurace Apache, PHP, MySQL.

Při aplikaci souborových ACL je důležité dát pozor i na první pohled nepatrné detajly. Například pokud uživateli web zakážeme číst soubor */etc/resolv.conf* nebude možné ve skriptovacím jazyce PHP používat doménový překlad.

Kompletní skript pro nastavení ACL ve sdílené šabloně se nalézá v příloze A.1.

5 Clusterové souborové systémy OCFS2 a GFS2

V podnikové sféře se pro dosažení vysoké dostupnosti služeb velice často používají HA clustery [23] řešené různým způsobem. Pro určité formy realizace HA clusteru jsou nezbytné clusterové souborové systémy [24]. Ty se od běžných souborových systémů liší tím, že umožňují současný přístup z více fyzických míst pro čtení i zápis. To sice přináší snížení výkonu souborového systému (je potřeba řešit zamykání a sdílený přístup ke zdrojům), ale nad jedním oddílem s clusterovým souborovým systémem pak může operovat najednou více serverů.

Více o roli HA clusterů s použitím clusterových souborových systémů popisuji v kapitole 6.

Na platformě GNU/Linuxu jsou prostřednictvím Linuxového jádra dostupné dva otevřené clusterové souborové systémy od významných hráčů na poli podnikových řešení. Jedná se o Oracle OCFS2 [25] a Red Hat GFS2 [26] souborové systémy. Obě tato řešení v této kapitole rozeberu co do zprovoznění pod Gentoo GNU/Linuxem, jejich výkonnostních testů a rozhodnutí, který ze souborových systémů by byl pro možnou implementaci HA clusteru na multiwebhostingové platformě nejvhodnější.

5.1 OCFS2 (Oracle Cluster File System 2)

Implementace clusterového souborového systému od firmy Oracle. Původně byl určen pro jejich vlastní databázová řešení. OCFS2 je šířen jako open source pod licencí GNU. Na scéně GNU/Linuxu se poprvé objevil v jádře 2.6.16, kde se nalézá a je aktualizován a udržován dodnes.

Některé vybrané vlastnosti OCFS2:

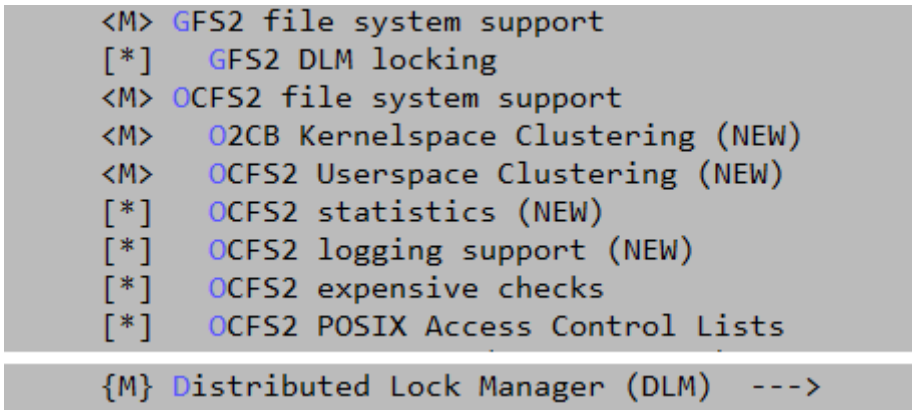
- Maximální velikost souboru 4 PB*.
- Maximální velikost oddílu 4 PB*.
- Maximální délka názvu souboru 255 znaků.
- Podpora standardních POSIX práv i ACL.
- Podpora žurnálu.
- Podpora zámků (DLM).

* *PB = PetaByte = 10^{15} bajtů.*

Kompletní přehled vlastností je pak v příslušné dokumentaci [25] nebo na wikipedii [27].

5.1.1 Instalace pod Gentoo GNU/Linuxem

Pro možnost využít clusterový souborový systém OCFS2 potřebujeme Linuxové jádro, které podporuje příslušný modul. V OpenVZ RHEL 6 jádře je tento modul dostupný. S výběrem OCFS2 se automaticky aktivuje i volba DLM, která je pro funkci OCFS2 nezbytná. Viz obrázek 5.1.



Obrázek 5.1 Konfigurace jádra pro souborové systémy OCFS2 a GFS2.

Dále je potřeba nainstalovat nástroje OCFS2 tools, které umožní například formátování diskového oddílu tímto souborovým systémem. Tyto nástroje jsou dostupné přímo v instalačním stromu portage Gentoo GNU/Linuxu. Instalace počítá s funkčním DRBD řešením na dvouuzlovém clusteru. DRBD popisuji v kapitole 6.

Odmaskování a instalace ocfs2-tools.

```
(gentoo)# echo "sys-fs/ocfs2-tools" >> \
/etc/portage/package.keywords
(gentoo)# emerge ocfs2-tools
```

Nahrávání potřebných modulů po startu systému.

```
(gentoo)# vi /etc/conf.d/modules
   modules="ocfs2 ocfs2_dlmfs configfs ocfs2_dlm"
```

Připojení potřebných pomocných struktur. Pravděpodobně bude potřeba restartovat počítač, příkazem *mount –a* se mi nepovedlo tyto struktury dopřipojit.

```
(gentoo)# vi /etc/fstab
   none          /sys/kernel/config          configfs
defaults                0 0

   none          /sys/kernel/dlm             ocfs2_dlmfs
defaults                0 0
```

Dále musíme upravit konfigurační soubory pro OCFS2. V konfiguraci je potřeba specifikovat název clusteru, IP adresy, porty a názvy jednotlivých uzlů clusteru. OCFS2 tedy počítá s clusterovým použitím, ale je možné provést nastavení i pro jeden uzel.

```
(gentoo)# vi /etc/conf.d/ocfs2
   OCFS2_CLUSTER="cluster1"
(gentoo)# mkdir /etc/ocfs2
(gentoo)# vi /etc/ocfs2/cluster.conf
   node:
      ip_port = 7777
      ip_address = 192.168.1.22
      number = 1
      name = pocitac1
      cluster = cluster1

   node:
      ip_port = 7777
      ip_address = 192.168.1.33
      number = 2
      name = pocitac2
      cluster = cluster1

   cluster:
      node_count = 2
      name = cluster1
```

Pak stačí již spustit OCFS2 démona a provést formátování příslušného oddílu souborovým systémem OCFS2. Níže formátuji DRBD oddíl *drbd0*.

```
(gentoo)# /etc/init.d/ocfs2 start
(gentoo)# mkfs.ocfs2 --force -b 4K -C 4K -N 4 -L ocfs2 -T mail \
   --fs-feature-level=max-features --verbos /dev/drbd0
```

Instalace OCFS2 pod Gentoo GNU/Linuxem je tedy velice jednoduchá a rychlá. Ze strany dostribuce i Linuxového jádra je OCFS2 přímo podporováno.

5.2 GFS2 (Flobal File System 2)

Implementace clusterového souborového systému od firmy Red Hat. Do GNU/Linuxu pronikl v Linuxovém jádře 2.6.19, kde je udržován a aktualizován dodnes. Stejně jako OCFS2, tak i GFS2 je šířen jako open source pod licencí GNU.

Některé vybrané vlastnosti GFS2:

- Maximální velikost souboru 8 EB*.
- Maximální velikost oddílu 8 EB*.

- Maximální délka názvu souboru 255 znaků.
- Podpora standardních POSIX práv i ACL.
- Podpora žurnálu.
- Podpora zámků (DLM a nolock).

[*] *EB = ExaByte = 10^{18} bajtů.*

Kompletní přehled vlastností je pak v příslušné dokumentaci [25] nebo na wikipedii [27].

5.2.1 Instalace pod Gentoo GNU/Linuxem

Pro možnost využít clusterový souborový systém GFS2 potřebujeme Linuxové jádro, které podporuje příslušný modul. V OpenVZ RHEL 6 jádře je tento modul dostupný. S výběrem GFS2 se automaticky aktivuje i volba DLM, která je pro funkci GFS2 v nezbytná. Viz obrázek 5.1 v kapitole 5.1.1.

Dále je potřeba nainstalovat nástroje, které umožní provoz GFS2 pod Gentoo GNU/Linuxem a formátování a připojení diskového oddílu tímto souborovým systémem. Zde jsem u GFS2 narazil na první problém. Ze strany distribuce nemá tento souborový systém oficiální podporu. Nelze tedy využít snadné instalace z oficiálního stromu portage jak tomu bylo u OCFS2. Instalace počítá s funkčním DRBD řešením na dvouuzlovém clusteru. DRBD popisuji v kapitole 6.

Při hledání jak do Gentoo GNU/Linuxu nainstalovat podporu GFS2 jsem narazil na overlay stromu portage, který obsahuje vše potřebné. Overlay je neoficiální instalační strom, který je možné použít pro instalaci neoficiálního software v Gentoo GNU/linuxu. Jedná se o overlay s názvem *wolf31o2*. Kromě využití portage overlaye bylo ale potřeba několika dodatečným úprav, aby bylo možno vše potřebné nainstalovat.

Přidání potřebné overlay.

```
(gentoo)# emerge -av layman
(gentoo)# echo "source /var/lib/layman/make.conf" >> \
   /etc/make.conf
(gentoo)# layman -L
(gentoo)# layman -a wolf3lo2
```

Úprava *sys-cluster/openais-0.80.6* pro korektní načtení patchů.

```
(gentoo)# vi openais-0.80.6.ebuild
ZMĚNIT src_unpack():
   src_unpack() {
      unpack ${A}
      cd "${S}"
      epatch "${FILESDIR}"/${P}-Makefile.inc.patch || die
```

```
        epatch "${FILESDIR}"/${P}-Makefile.patch || die
    }
```

Úprava *sys-cluster/dlm-lib-2.03.10*, provádí se v něm chybná detekce zdrojových kódů jádra.

```
(gentoo)# vi sys-cluster/dlm-lib-2.03.10
ZMĚNIT --kernel_src=${KERNEL_DIR}:
   --kernel_src="${D}/usr/src/linux"
```

Úprava *sys-cluster/ccs-2.03.10*.

```
(gentoo)# vi sys-cluster/ccs-2.03.10
ODSTRANIT:
   epatch "${FILESDIR}"/ccs-2.03.09-mkostemp.patch || die
```

Úprava *sys-cluster/fence-2.03.10*.

```
(gentoo)# vi sys-cluster/fence-2.03.10
ZMĚNIT =sys-cluster/openais-0.80.3*:
   >=sys-cluster/openais-0.80.3
ODSTRANIT:
   epatch "${FILESDIR}"/fence-2.03.09-ipmi_fix_shell.patch || die
   epatch "${FILESDIR}"/fence-2.03.09 \
     ipmi_lan_timeout_adjusted_and_configurable.patch || die
```

Úprava *sys-cluster/cman-2.03.10*.

```
(gentoo)# vi sys-cluster/cman-2.03.10
ZMĚNIT =sys-cluster/openais-0.80.3*:
   >=sys-cluster/openais-0.80.3
ODSTRANIT:
   epatch "${FILESDIR}"/${P}-add_votes_to_transition_message.patch
|| die
   epatch "${FILESDIR}"/${P}-
fix_signatures_of_cman_get_privdata_and_cman_set_privdata.patch ||
die
```

Pro kompilaci *sys-fs/gfs2-2.03.10* je potřeba mít v systému knihovnu *dev-libs/libvolume_id.* Tu je před instalací nutné odmaskovat a povolit pro architekturu 64 bitového prostředí.

```
(gentoo)# echo "dev-libs/libvolume_id" >> \
   /etc/portage/package.keywords
(gentoo)# vi /etc/make.conf
   KEYWORDS="~x86-fbsd amd64"
```

Dále je nutné upravit některé soubory jádra, jinak instalace GFS2 nástrojů skončí s chybou.

```
(gentoo)# vi /usr/src/linux/include/asm-generic/int-ll64.h
ZMĚNIT #include <asm/bitsperlong.h>:
   #include <asm-generic/bitsperlong.h>
```

```
(gentoo)# vi /usr/src/linux/include/linux/gfs2_ondisk.h
ZMĚNIT gfs2_quota:
   struct gfs2_quota {
      __be64 qu_limit;
      __be64 qu_warn;
      __be64 qu_value;
      __be32 qu_ll_next;
      __u8 qu_reserved[64];
   };
```

Při instalaci GFS2 je nutné si pohlídat, aby nedocházelo k instalaci balíků různých verzí. Vše musí být ve stejné verzi (2.03.10).

```
(gentoo)# echo "=sys-cluster/openais-0.80.6" >> \
   /etc/portage/package.keywords
(gentoo)# echo "=sys-cluster/ccs-2.03.10" >> \
   /etc/portage/package.keywords
(gentoo)# echo "=sys-cluster/cman-2.03.10" >> \
   /etc/portage/package.keywords
(gentoo)# echo "=sys-cluster/cman-lib-2.03.10" >> \
   /etc/portage/package.keywords
(gentoo)# echo "=sys-cluster/dlm-2.03.10" >> \
   /etc/portage/package.keywords
(gentoo)# echo "=sys-cluster/dlm-lib-2.03.10" >> \
   /etc/portage/package.keyword
(gentoo)# echo "=sys-cluster/dlm-lib-2.03.10" >> \
   /etc/portage/package.keywords
(gentoo)# echo "=sys-cluster/fence-2.03.10" >> \
   /etc/portage/package.keywords

(gentoo)# emerge -av gfs2 libvolume_id
```

Konfigurační soubor se nalézá v */etc/cluster/cluster.conf* a má formát XML souboru.

```
<?xml version="1.0"?>
<cluster alias="cluster-setup" config_version="1" name="cluster-setup">
  <rm log_level="4"/>
  <fence_daemon clean_start="1" post_fail_delay="0" post_join_delay="3"/>
  <clusternodes>
    <clusternode name="node1" nodeid="1" votes="1">
      <fence>
        <method name="2">
          <device name="LastResortNode01"/>
        </method>
      </fence>
    </clusternode>
    <clusternode name="node2" nodeid="2" votes="1">
      <fence>
        <method name="2">
          <device name="LastResortNode02"/>
        </method>
      </fence>
    </clusternode>
  </clusternodes>
  <cman expected_votes="1" two_node="1"/>
  <fencedevices>
    <fencedevice agent="fence_manual" name="LastResortNode01" nodename="node1"/>
    <fencedevice agent="fence_manual" name="LastResortNode02" nodename="node2"/>
  </fencedevices>
  <rm/>
  <totem consensus="4800" join="60" token="10000" token_retransmits_before_loss_const="20"/>
</cluster>
```

Vytvoření souborového systému GFS2 pak probíhá následujícím způsobem.

```
(gentoo)# mkfs.gfs2 -p lock_dlm -t cluster-setup:res0 \
   /dev/drbd0 -j 2
```

GFS2 standardně počítá s minimálně dvouuzlovým clusterem. Je však možné při vytváření souborového systému použít i volbu *–p nolock* pro formátování lokálního zařízení jednoho uzlu.

Jak je vidět, instalace GFS2 pod Gentoo GNU/Linux je dost složitá. V době kdy jsem toto řešení testoval neexistovala v oficiálním instalačním stromu portage podpora pro GFS2, bylo tedy nutné využít neoficiálního zdroje. Navíc je potřeba provést řadu dalších úprav, aby bylo možno instalovat vše potřebné.

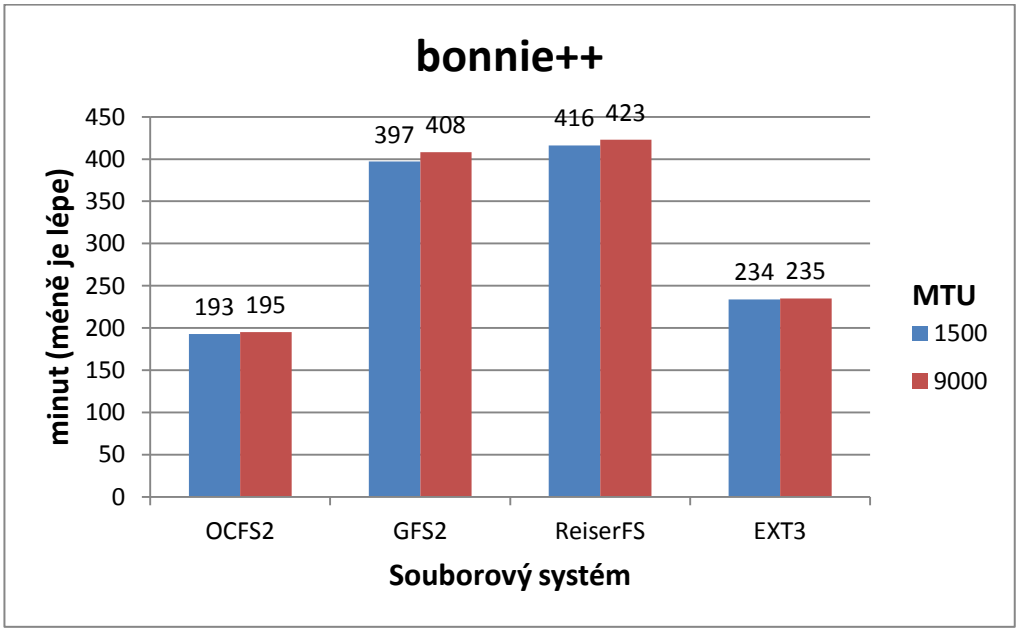
5.3 Výkonové testy

S funkčními instalacemi clusterových souborových systémů je možné otestovat jejich výkon a stabilitu. Všechny testy jsem prováděl na dvou hardwarově shodných serverech třídy PC. Servery obsahovaly procesor Intel Xeon architektury i7, 12 GB RAM v tříkanálovém režimu, gigabitovou síťovou kartu Intel a dva 7200 otáčkové disky zapojené do diskového pole RAID typu 1. Tyto dva servery jsem nakonfiguroval pro podporu testovaných souborových systémů a zapojil jsem je do DRBD clusteru.

Testování clusterových souborových systémů pak probíhalo na DRBD zařízení, užitém pro sdílení diskového oddílu mezi oběma uzly clusteru. Testy jsem prováděl vždy na jednom z uzlů clusteru, případné změny v souborovém systému jsou pak přenášeny a synchronizovány s druhým uzlem logikou DRBD po vzájemné ethernetové síti mezi uzly. Jako data byla použita záloha jednoho z produkčních webhostingových serverů, která obsahovala několik stovek GB webových dat, tedy převážně velké množství malých souborů.

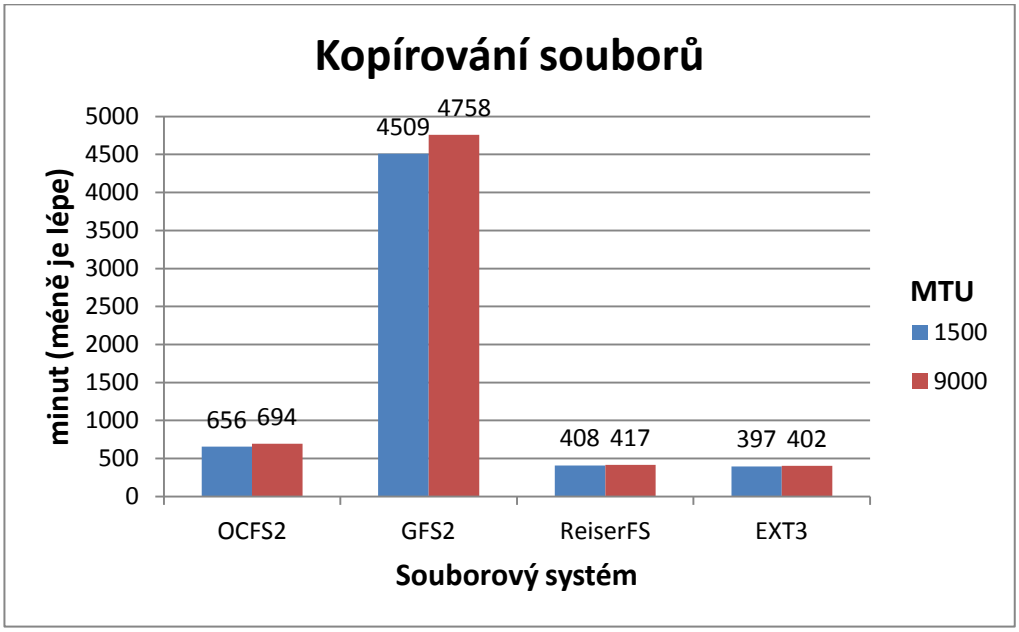
Výsledky jsou uváděny jako průměry ze dvou opakovaných měření. Jelikož se dá očekávat, že clusterové souborové systémy budou méně výkonné, než standardní souborové systémy, protože jejich architektura je složitější a počítá s konkurenčním přístupem k datům, rozhodl jsem se zařadit do testování také standardní neclusterové souborové systémy ReiserFS [28] a EXT3 [29]. Vzhledem k synchronizacu uzlů clusteru přes ethernetovou síť jsem se rozhodl zahrnout do testů také použití rozličného *MTU*. Konkrétně jsem testoval standardní ethernetové MTU o velikosti 1500 bajtů a dále pak tzv. jumbo rámec [30] o velikosti 9000 bajtů. U všech testů jsem prováděl měření délky běhu testu systémovým nástrojem *time*.

Prvním z testů bylo měření výkonu nástrojem *bonnie++* [31]. Clusterovému souborovému systému OCFS2 se podařilo zvítězit i nad standardními souborovými systémy, což je dáno pravděpodobně použitým nastavením testu. Výsledky jsou na obrázku 5.2

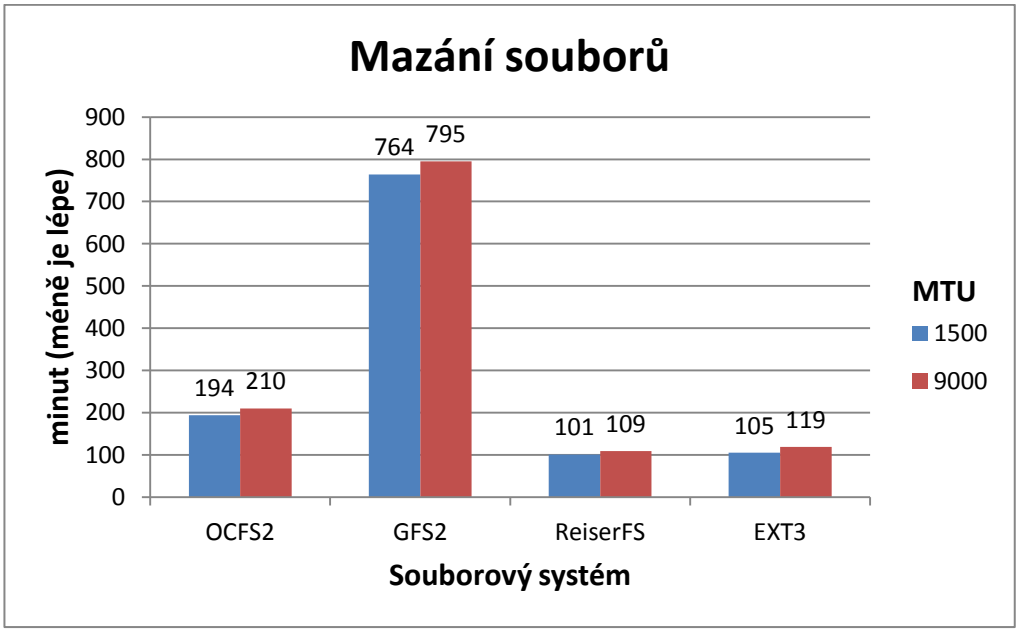


Obrázek 5.2 Výkon souborových systémů v bonnie++.

Další dva testy zastupují standardní operace prováděné s daty – kopírování a mazání. Překvapením je clusterový souborový systém GFS2, který je 7x pomalejší než jeho konkurenční kolega OCFS2 a 10x pomalejší než běžné souborové systémy. Výsledky jsou na obrázcích 5.3 a 5.4.

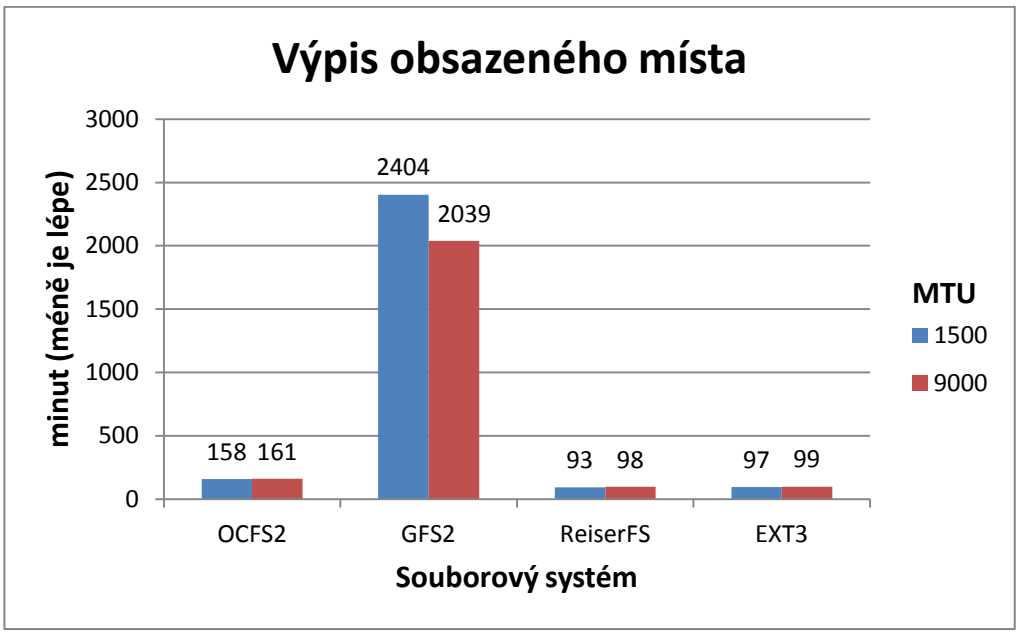


Obrázek 5.3 Výkon souborových systémů při kopírování.

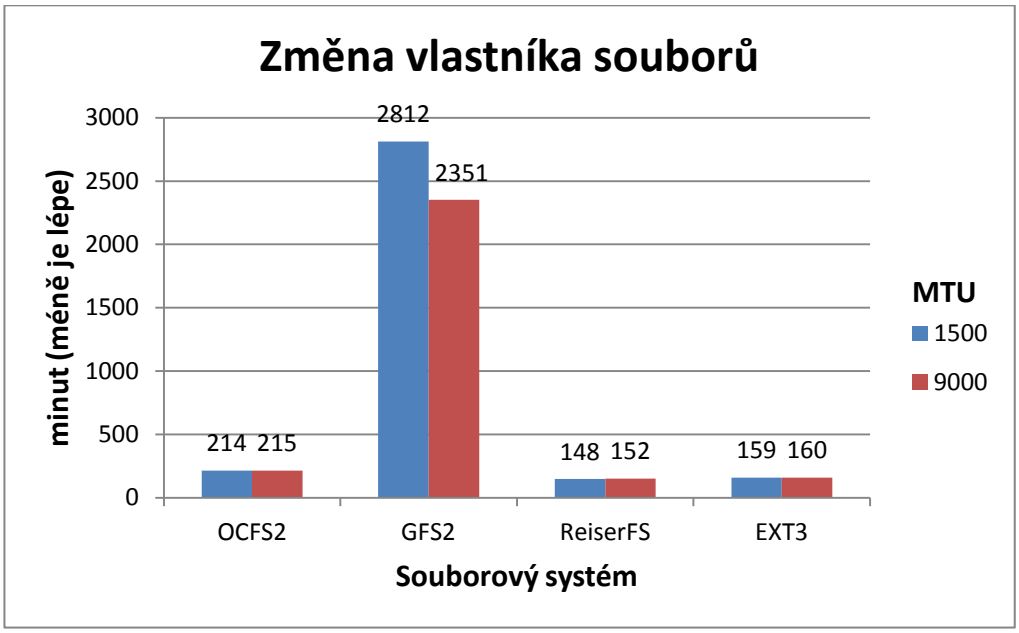


Obrázek 5.4 Výkon souborových systémů při mazání.

Jako další dva testy jsem se rozhodl zvolit nějaké praktické akce prováděné nad souborovým systémem. Zvolil jsem výpis obsazeného místa adresáři *du –hs /data/** a rekurzivní změnu vlastníka souborů *chown –R /data*. Opět se opakuje drastický výkonový propad GFS2. OCFS2 je zhruba 1,5x pomalejší, než běžné souborové systémy. Výsledky jsou na obrázcích 5.5 a 5.6.

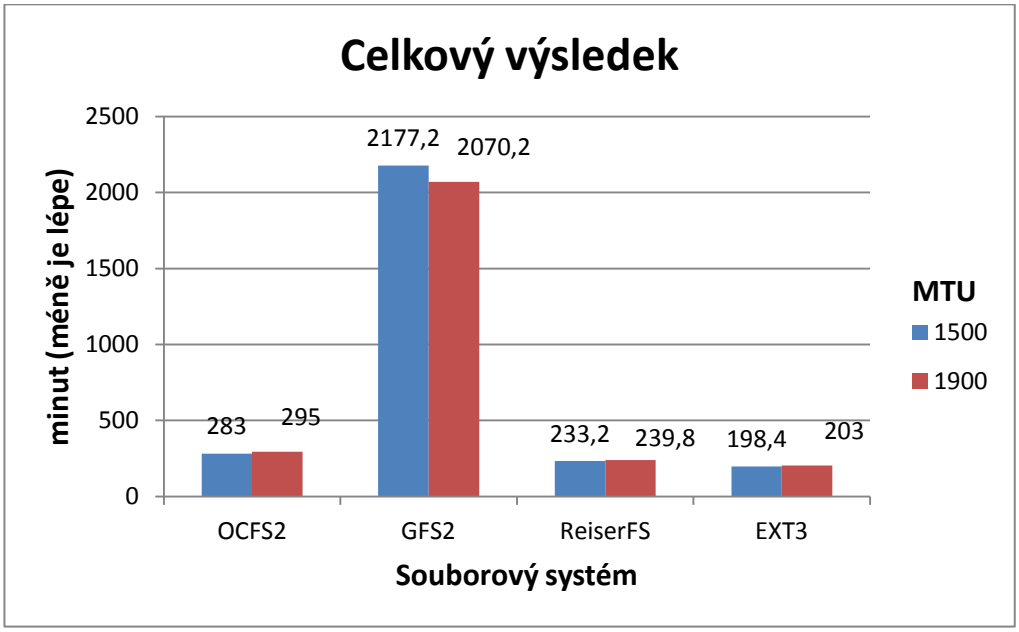


Obrázek 5.5 Výkon souborových systémů při výpisu volného místa.

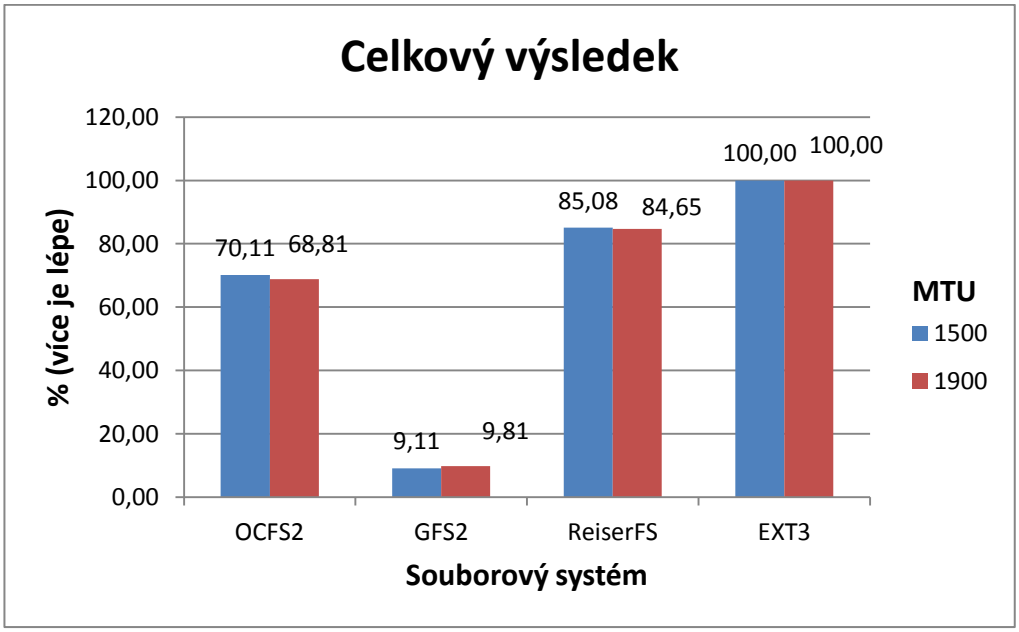


Obrázek 5.6 Výkon souborových systémů při změně vlastníka souborů.

Na obrázku 5.7 je shrnutí všech provedených testů průměrem. Obrázek 5.8 přináší procentuální shrnutí výsledků, kde jako 100% je zvolen souborový systém, který dosáhl nejvyššího průměrného výkonu – EXT3.



Obrázek 5.7 Přehled výkonu souborových systémů.



Obrázek 5.8 Přehled výkonu souborových systémů.

GFS2 tedy naprosto propadlo. Na svého konkurenta z clusterových souborových systémů OCFS2 ztrácí 61%. Oproti nejvýkonnějšímu EXT3 pak poskytuje jen desetinový výkon. OCFS2 proti EXT3 ztrácí zhruba 30%. Rozdíl mezi standardními souborovými systémy EXT3 a ReiserFS je přibližně 15% ve prospěch EXT3. Rozdíl v testovaných velikostech MTU se neprojevil. Odchylka mezi 1500 a 9000 bajty je maximálně 2%. To je pravděpodobně dáno povahou dat, která představují velké množství malých souborů. Zvětšování MTU má pak význam jen pro přenos velkých souborů. Během testování jsem nezaznamenal žádné problémy se stabilitou testovaných prostředí.

5.4 Výběr clusterového souborového systému

Pro výběr nejvhodnějšího clusterového souborového systému jsem se rozhodl zohlednit následující kritéria.

- Výkon souborového systému.
- Stabilita souborového systému.
- Dostupná funkcionalita souborového systému.
- Snadnost instalace do serverového prostředí.

Na základě seznámení se s clusterovými souborovými systémy, jejich instalací a otestováním v serverovém prostředí jsem se rozhodl zvolit OCFS2 od Oraclu. GFS2 souborový systém jednak ve výkonových testech naprosto propadl a za druhé jeho instalace pod Gentoo GNU/Linuxem je velice složitá a nepovažuji ji za vhodnou pro produkční užití. Ze specifických požadavků na funkcionalitu jsou kladeny požadavky na podporu souborových ACL, což splňují oba souborové systémy.

V kapitole 6 se věnuji zajištění dostupnosti multiwebhostingové platformy a tím tedy i HA clusterům však nakonec nepoužiji žádný clusterový souborový systém, ale jiné řešení HA clusteru. Důvod je takový, že byť jsem se během testování nesetkal u OCFS2 s problémy se stabilitou, tak ze strany firmy byl tento souborový systém používán na sdíleném webhostingu a neprojevoval se dlouhodobě patřičně stabilně.

6 Zajištění dostupnosti multiwebhostingu

Multiwebhostingová platforma bude poskytovat své služby velkému množství zákazníků. Každý z nich navíc na takové platformě bude hostovat více domén a je tedy nutné se kromě poskytovaného výkonu, možností nastavení a výhodné ceny zabývat také otázkou dostupnosti služby. Je potřeba navrhnout takové řešení, aby byl výpadek buď vyloučen nebo doba pro jeho překlenutí byla minimální. Není tedy možné, aby se několik stovek zákazníků a desítky domén každého z nich kvůli výpadku hostitelského serveru staly na delší dobu nedostupné. Nejběžnější řešení pro zajištění dostupnosti je záloha redundantním hardwarem a tzv. High Availability (dále HA) clustery.

Budu se zde zabývat zajištěním dostupnosti v případě selhání hardwarových prostředků. Softwarové poškození se pak řeší standardně pravidelným zálohováním zákaznických dat i struktur hostitelských systémů. Rovněž se zde nebudu zabývat řešením diskových polí RAID která předchází výpadkům dostupnosti diskových dat, jelikož se počítá s tím, že takové uspořádání disků je na produkčních serverech použito vždy a jeho implementace je triviální záležitostí.

V této kapitole proberu několik technik, jak je možné zajistit vysokou dostupnost služby a minimalizovat případné výpadky serverů. Bude zde zmíněno clusterové řešení DRBD a jeho různé režimy operace. Rozeberu zde také možnost rozdělit hostitelské prostředí na aplikační a úložný server, tedy SAN řešení. V závěru kapitoly pak navrhnu nejvhodnější řešení pro platformu multiwebhostingu.

6.1 Redundance hardwaru

Nejjednodušší způsob jak zajistit dostupnost služby je mít k dispozici nadbytečný hardware, který v případě selhání původního převezme jeho funkci. Na straně serveru je to pak řešeno nejčastěji způsobem kompletního náhradního serveru, do kterého se v případě výpadku pouze přesunou disky s daty původního serveru. Takovéto řešení sebou přináší následující aspekty, které je třeba zohlednit.

- **Cena** – mít fyzickou zálohu pro většinu nebo všechny servery je především pro malé firmy finančně nemožné. Často je pak záložních serverů jen několik, protože se nepočítá s tím, že dojde k vícenásobnému výpadku kompletních serverů.
- **Umístění záložního hardwaru** – aby bylo možno rychle přesunout disky z havarovaného serveru musí být záložní server instalován poblíž toho původního a být součástí síťové struktury. To zabírá místo a přidává další kabeláž.
- **Homogenita hardwaru** – aby operační systém havarovaného serveru mohl rychle nastartovat a služba byla obnovena, je potřeba, aby hardwarové prostředí záložního serveru co nejvíce odpovídalo tomu původnímu. Problém může nastat například s různými diskovými šachtami, s jádrem, které nepodporuje např. síťovou kartu základní desky záložního serveru apod.

- **Doba výpadku** – spuštění hardwarové zálohy může být časově náročnější, než softwarové řešení na úrovni HA clusteru, který je schopen reagovat rychleji.
- **Neřeší vše** – ani toto řešení není absolutní, byť se zdá, že mít k dispozici kompletní náhradní server vyřeší každý problém s hardwarem. Například v případě selhání disku mohou nastat další komplikace. I přes provoz na diskových polích typu RAID, která chrání data proti selhání disku se může stát, že se data nějakým způsobem poruší a je potřeba je obnovit ze softwarové zálohy. Taková operace je v případě velkého množství dat časově velmi náročná (hodiny) a data nebudou úplně aktuální, jelikož typicky velké zálohy se provádí maximálně v denních cyklech.

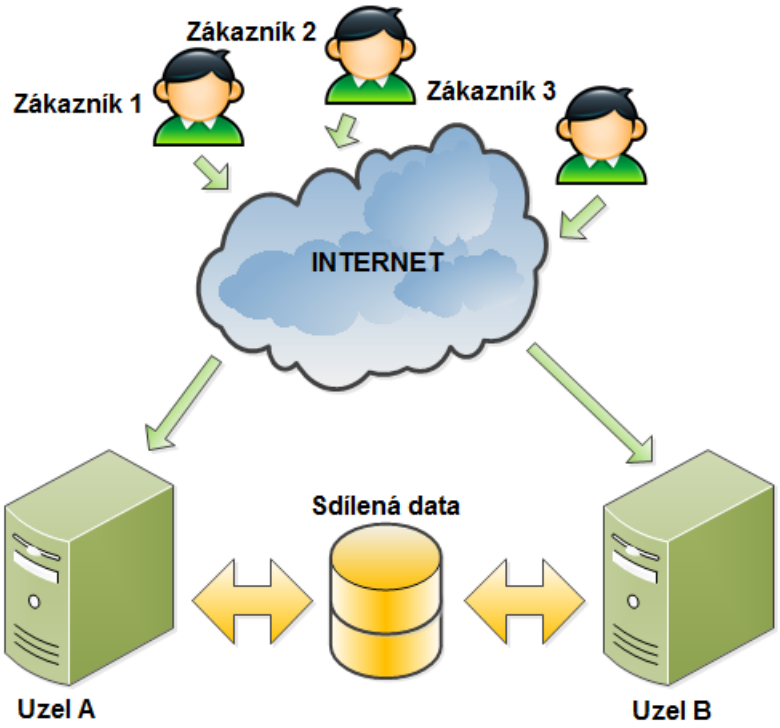
Zálohování nadbytečným hardwarem je tedy dobrá pojistka pro výpadek kompletního serveru, ale není to vhodné řešení přímo pro zajištění dostupnosti služby. Je tedy potřeba přijít s jiným řešením, které si poradí s výpadkem efektivněji, zatímco se bude řešit problém s havarovaným hardwarem.

6.2 HA cluster

HA cluster nebo také cluster s vysokou dostupností je takové zapojení serverů poskytujících službu, které v případě selhání jednoho nebo více serverů umožňuje službu poskytovat i nadále s minimálním výpadkem dostupnosti. V clusterové terminologii se pak jednotlivé servery účastnící se clusteru označují jako uzly clusteru.

V případě selhání některého uzlu je nutné, aby jiný uzel převzal činnosti toho původního. Aby bylo možno minimalizovat dobu, kdy jiný uzel převezme práci za havarovaný uzel, je nutné zajistit, aby všechny uzly clusteru, které se mají vzájemně krýt, měly mezi sebou dostupná data a vhodné softwarové prostředí (databázový server bude těžko fungovat jako webový). Princip HA clusteru je nastíněn na obrázku 6.1. Zákaznící nemají tušení, kde je jejich služba hostována ani to pro ně není podstatné. Na straně firmy pak službu poskytuje více serverů, které mezi sebou sdílí data. Pokud jeden z uzlů vypadne, druhý může převzít rychle jeho činnosti, protože má k dispozici data vypadlého uzlu.

Pro převzetí služby za havarovaný uzel (tato operace se nazývá jako *failover*) se implementují různé techniky od plně automatizovaného procesu, kdy se uzly musí navzájem prověřovat, zda jsou funkční (*heartbeat*), až po ruční zásah, o kterém je administrátor například informován pouze formou nějakého monitoringu dostupnosti. V případě zjištění výpadku některého uzlu, je pak nutné provést určitou sadu kroků, která vede k obnovení služby na jiném uzlu (převzetí IP adresy havarovaného uzlu, načtení vhodné konfigurace pro démony realizující službu, apod.).



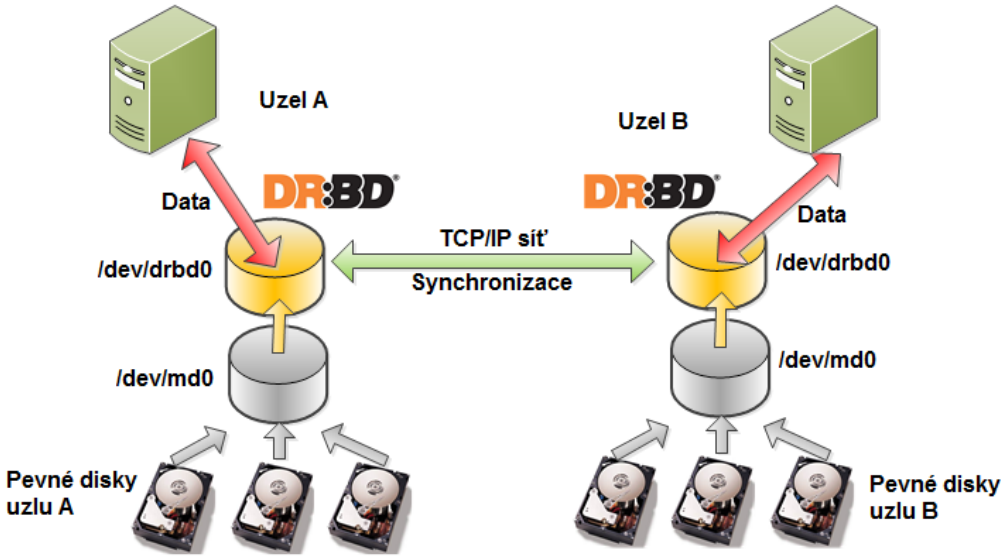
Obrázek 6.1 Princip sdílení dat mezi uzly clusteru

Z hlediska toho, které uzly vyřizují požadavky na službu se pak HA clustery dělí na dva základní režimy operace.

- **Aktivní / Aktivní** – pro realizaci služby je využíváno všech nebo většiny uzlů. To je výhodné, protože všechny dostupné servery jsou využity. Nevýhoda je, že je potřeba volit vhodnou výkonovou dimenzaci serverů, jelikož náhradní uzel pak musí kromě svých vlastních požadavků vyřizovat i požadavky uzlu vypadlého. V případě společného sdíleného oddílu s daty je pak také nutné implementovat clusterový souborový systém, který umožňuje paralelní přístup.
- **Aktivní / Pasivní –** uzel clusteru, který vyřizuje požadavky na službu je zálohován jiným uzlem, který však požadavky nevyřizuje, pouze v případě výpadku aktivního uzlu převezme jeho činnost. Toto řešení je podobné hardwarové redundanci, ale díky clusteru je možné dosáhnout kratších výpadků. Pasivní uzel si jen připojí oddíl s daty, převezme si IP adresu vypadlého uzlu a může okamžitě začít vyřizovat požadavky.

6.3 DRBD

DRBD - Distributed Replicated Block Device [32] nebo-li česky distribuované replikované blokové zařízení je otevřená technologie, která se používá pro implementaci HA clusteru v praxi.



Obrázek 6.2 Princip funkce DRBD.

Na obrázku 6.2 je znázorněn princip funkce DRBD. Principielně se dá tato technologie představit jako zavedení diskového pole RAID 1 po síti. Uzly používající DRBD pak tedy mají k dispozici blokové zařízení, které je synchronizováno po síti. To řeší základní požadavek HA clusteru na společná data mezi uzly. Samotná data jsou umístěna na standardním disku nebo lépe RAID poli disků uzlů. Oddíl s daty, který má pak sloužit pro sdílené využití clusterem, je transformován na DRBD zařízení. Aplikační logika DRBD potom zajistí provázání a synchronizaci se stejným DRBD zařízením na protějším uzlu clusteru přes počítačovou síť. Nezávislých diskových oddílů, které mají být použity jako DRBD zařízení může být samozřejmě více a ta mohou být nakonfigurována v různých režimech.

Je důležité rozlišovat mezi režimy HA clusteru specifikovanými v kapitole 6.2 a mezi režimy poskytovanými technologií DRBD, byť na první pohled jsou si podobné. Režim DRBD se vždy týká pouze konkrétního blokového zařízení na uzlu, které je synchronizováno po síti se zařízením na uzlu protějším rovněž v určitém režimu. Každý uzel může mít těchto zařízení více a v různých režimech.

6.3.1 DRBD režim Primary – Primary

V tomto režimu je DRBD zařízení synchronizované mezi uzly clusteru možné připojit z obou uzlů a číst a zapisovat do něj data. Změny v datech provedene jedním uzlem má tedy druhý uzel okamžitě k dispozici. Takové zařízení je možné použít pro současný přístup ke sdíleným datům. To

sebou ovšem přináší specifické nároky na použitý souborový systém. Jelikož je potřeba řešit paralelní přístup (DRBD zařízení je vícenásobně připojeno), je nutné použít clusterový souborový systém, který umožní zachovat konzistenci dat.

6.3.2 DRBD režim Primary – Secondary

V tomto režimu je DRBD zařízení také přítomné a synchronizované na obou uzlech clusteru, ale přístup k datům má vždy pouze jeden z uzlů a to ten, na kterém je zařízení v režimu primary. DRBD je navíc tak striktní, že zařízení v režimu secondary nemůže být připojeno ani pro čtení. Tento režim tedy sice neumožňuje, aby oba uzly zároveň pracovaly nad stejným sdíleným zařízením, ale umožňuje použít standardní souborový systém typu EXT nebo ReiserFS. Zároveň je zachována výhoda, že data jsou dostupná i na jiném uzlu, který je v případě výpadku může použít. Tento režim tedy nevylučuje možnost použití v HA clusteru typu Aktivní/Aktivní, pouze neumožňuje použít jeden sdílený oddíl.

6.3.3 Instalace pod Gentoo GNU/Linuxem

Ve svých začátcích vyžadovalo DRBD dodatečnou kompilaci zvláštního jaderného modulu, aby bylo možno použít pod systémem speciální bloková zařízení. Bohužel takto nebylo možné zkompilovat modul pro každou verzi jádra a pro silně patchovaná jádra virtualizačních řešení jako je Xen a OpenVZ tak vznikala spousta problémů, jak modul vlastně do jádra dostat. To se vyřešilo s příchodem jádra verze 2.6.33, kdy byl do něj modul oficiálně zahrnut a je tam udržován a pravidelně aktualizován dodnes, jelikož DRBD se stalo standardem pro HA řešení. OpenVZ sice používá RHEL6 jádro založené na verzi 2.6.32, ale Red Hat do tohoto jádra modul DRBD rovněž zpětně implementoval a není tedy problém jej použít.

Pod Gentoo GNU/Linuxem pak jen stačí nainstalovat správcovské nástroje *sys-cluster/drbd*. Je dobré používat vždy nástroje stejné verze, jako je modul v jádře.

```
(gentoo)# emerge -av drbd
```

Konfigurace se pak nalézá v souboru */etc/drbd.conf*. Níže uvádím jen část konfigurace pro definici jednoho z DRBD zařízení *r1*. V definici je specifikován název DRBD zařízení *drbd1*, který fyzický disk se použije (*md3)* a IP adresy uzlů clusteru pro synchronizaci dat.

```
(gentoo)# vi /etc/drbd.conf

   resource r1 {
      device     /dev/drbd1;
      disk       /dev/md3;
      meta-disk  internal;
      on uzel1 {
         address    192.168.1.11:7788;
      }
```

```
        on uzel2 {
           address   192.168.1.12:7788;
        }
    }
```

DRBD zařízení se pak aktivuje a spustí se jeho synchronizace.

```
(gentoo)# drbdadm up r1
(gentoo)# drbdadm -- --overwrite-data-of-peer primary r1
```

Stav dostupných DRBD zařízení je možné sledovat v */proc* struktuře.

```
(gentoo)# cat /proc/drbd

1: cs:Connected ro:Primary/Secondary ds:UpToDate/UpToDate C r-----
   ns:43462520 nr:3063436 dw:46353924 dr:7576574 al:9652 bm:35
lo:0 pe:0 ua:0 ap:0 ep:1 wo:b oos:0
```

Vytvořené DRBD zařízení se může naformátovat a připojit jako běžný diskový oddíl.

```
(gentoo)# mkfs.ext4 /dev/drbd1
(gentoo)# mount /dev/drbd1 /uzel1
```

6.4 SAN

Pro implementaci zajištění dostupnosti služby na platformě multiwebhostingu jsem se zabýval také otázkou rozdělení hostitelského serveru na aplikační a úložnou část. K tomuto řešení je navržena technologie SAN.

Na obrázku 6.3 je znázorněn princip této technologie. Ta spočívá v centralizaci datového uložiště a přístupu aplikačních serverů přes síť. Nevyužívá se tedy ukládání dat na konkrétních serverech, ale na vyhrazeném síťovém uložišti. Centralizované uložiště sebou vždy přináší řadu výhod i nevýhod.

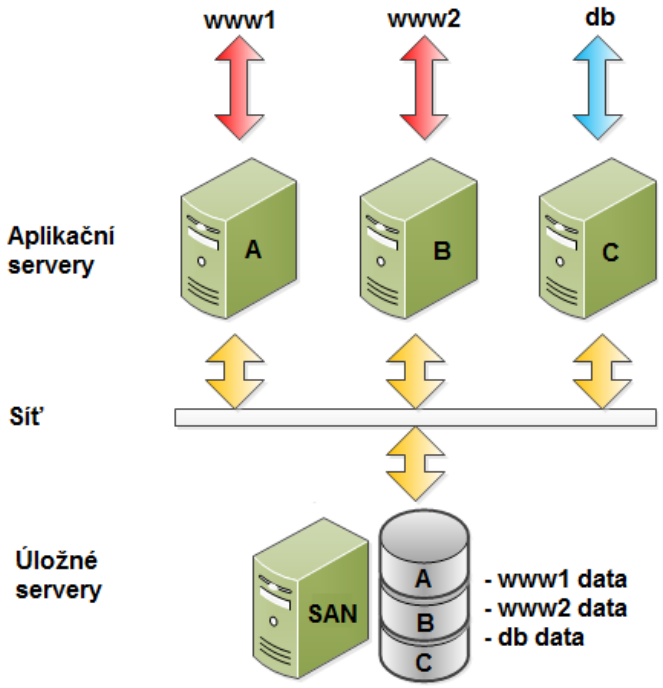
Výhody:

- Data jsou dostupné přes síť libovolnému serveru.
- Možnost optimalizovat SAN server přímo pro diskové operace.
- Přenesení problematiky sdílených dat clusteru z aplikačních serverů.

Nevýhody:

- Výpadek SAN způsobí nedostupnost dat všem aplikačním serverům.
- Náročnost na síťovou propustnost.

- Nutnost řešit problematiku sdílených dat pro clustery na SAN serveru.



Obrázek 6.3 Použití SAN.

SAN exportuje své diskové zdroje po síti na aplikační servery. Tento export probíhá ve formě blokových zařízení, která si pak mohou aplikační servery běžně připojit pro použití. Má-li být exportované zařízení sdíleno více aplikačními servery, je potřeba řešit problematiku paralelního přístupu a clusterový souborový systém. Při využití SAN řešení není dále na aplikačních serverech v clusteru nutné řešit dostupnost dat vzájemně mezi uzly. Uzel si může vždy prostřednictvím SAN importovat potřebný oddíl s daty.

6.4.1 iSCSI

Pokud chceme tedy implementovat SAN model, musíme použít vhodnou technologii, která umožňuje import/export blokových zařízení mezi úložnými a aplikačními servery. Nejdostupnější, nejlevnější a otevřené řešení pro GNU/Linux je iSCSI [33]. Tato technologie funguje na IP síti a je tedy pro její realizaci možné použít dostupnou a levnou ethernetovou síť.

iSCSI sestává z podpory Linuxového jádra a serverového a klientského softwaru pro export/import blokových zařízení. iSCSI server se nazývá *iSCSI target* a klient *iSCSI initiator*. Zkoumal jsem, jak je možno tuto technologii použít pod Gentoo GNU/Linuxem, který používám pro prostředí multiwebhostingové platformy.

iSCSI target je dostupný jako software *sys-block/iscsitarget.*

```
(gentoo)# emerge -av iscsitarget
```

Po instalaci je potřeba provést konfiguraci exportovaných oddílů. Nebudu zde zabíhat do detajlů, samozný konfigurační soubor je bohatě komentován. Pro příklad je níže uveden export dvou oddílů */dev/md4* a */dev/md2*. Každá exportovaná položka musí být opatřena unikátním názvem.

```
(gentoo)# vi /etc/ietd.conf
   Target iqn.2012-01.local.san01:md4
      Lun 0 Path=/dev/md4,Type=fileio
      MaxConnections 2

   Target iqn.2012-01.local.san01:md2ro
      Lun 1 Path=/dev/md2,Type=fileio,IOMode=ro
       MaxConnections 2
```

Pokud poté službu iscsitargetu nastartujeme, je možné exportované oddíly využívat po síti.

```
(gentoo)# /etc/ietd start
```

Dostupné oddíly a navázané iSCSI relace je pak možné sledovat přes */proc* strukturu.

```
(gentoo)# cat /proc/net/iet/volume
(gentoo)# cat /proc/net/iet/session
```

iSCSI initiator je dostupný jako software *sys-block/open-iscsi*.

```
(gentoo)# emerge -av open-iscsi
```

Po jeho instalaci je možné na daném targetu vyhledat a zobrazit dostupné exportované oddíly.

```
(gentoo)# iscsiadm -m discovery -t st -p 172.16.0.1
(gentoo)# iscsiadm -m node
```

Dostupné oddíly je pak možné zpřístupnit jako blokové zařízení pod klientem nebo jej odebrat.

```
(gentoo)# iscsiadm -m node -T iqn.2012-01.local.san01:md4 -l
(gentoo)# iscsiadm -m node -T iqn.2012-01.local.san01:md4 -u
```

Musel jsem se také zabývat výkonností tohoto řešení. Na gigabitové síti se mi nepodařilo dosáhnout rychlejších výsledků ani přes různé nalezené optimalizace nastavení sítě, než je 400 Mb/s. Problémy se stabilitou jsem nezaznamenal.

6.5 Výběr řešení pro zajištění dostupnosti multiwebhostingové platformy

Původně jsem počítal s návrhem řešení postaveným na HA clusteru v režimu Aktivní/Aktivní. Mezi jednotlivými hostitelskými servery multiwebhostingu by byl přes technologii DRBD sdílen

datový oddíl a servery by pracovaly vždy po dvojicích. V případě výpadku jednoho ze serverů by druhý uzel pouze spustil kontejnery, které se staly nedostupné vlivem výpadku prvního uzlu. Takové řešení by však vyžadovalo použít clusterový souborový systém GFS2 nebo OCFS2. GFS2 jsem zavrhl z důvodů velice složité instalace pod Gentoo GNU/Linuxem a špatného výkonu. OCFS2 se bohužel ve firemním prostředí dlouhodobě neosvědčil zase svou stabilitou. Na serverové farmě firmy byl OCFS2 používán asi ve dvouročním provozu na produkčních webhostingových serverech, přičemž docházelo pravidelně k několika výpadkům serverů během týdne, které se nepodařilo vyřešit. Po zamrznutí serveru a jeho tvrdém restartu byl navíc souborový systém často dost porušený a bývalo i nutné použít obnovu dat ze zálohy. To bylo časově velice náročné, navíc obnovená data nebyla naprosto aktuální. Z tohoto důvodu jsem po konzultaci s firmou odstoupil od použití clusterového souborového systému.

Pokud se jedná o otázku rozdělení aplikačních a úložných serverů, tedy implementaci SAN, zde jsem se rozhodl pro klasické řešení bez SAN. Ze softwarového hlediska by nebylo tak velkým problémem SAN implementovat, nicméně u iSCSI se mi nepodařilo dosáhnout dostatečné síťové propustnosti, což by se dalo řešit například zvláštním propojení každého z aplikačních serverů se SAN serverem na vyhrazených síťových kartách a bondingem síťových karet, ale takové řešení je již více nákladné, navíc je potřeba myslet i na fyzickou implementaci v serverovně. Také rozhodnutí používat servery pouze v párech přispívá pro myšlenku nepoužívat SAN, ale zprovoznit vždy jen vhodným způsobem DRBD mezi dvojicí multiwebhostingových hostitelů.

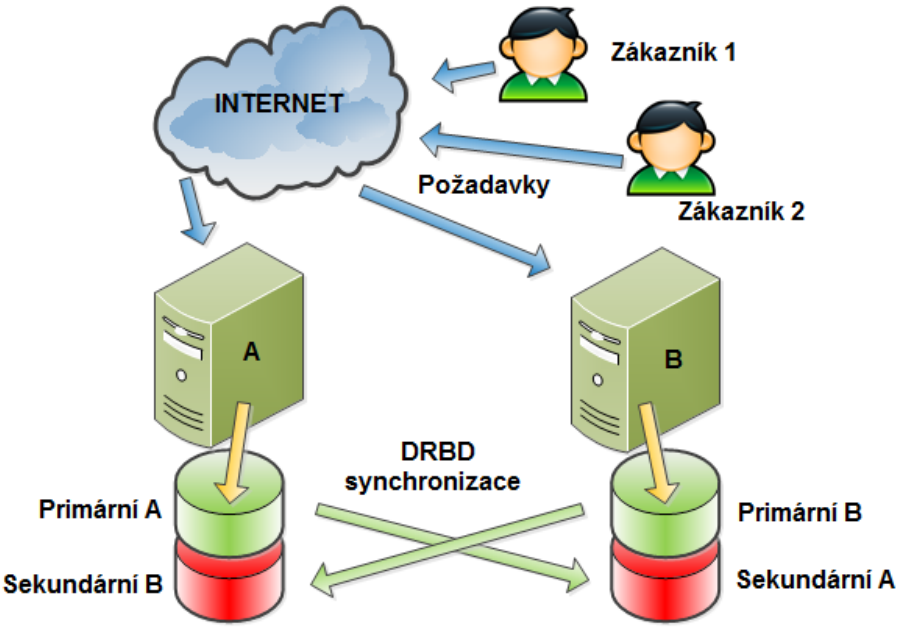
Pro zajištění vysoké dostupnosti HA clusterem jsem tedy konečně definoval tato kritéria:

- Hostitelé budou pracovat v párech.
- Každý z hostitelů musí být aktivním uzlem clusteru.
- Bez použití SAN.
- Bez použití clusterových souborových systémů.

6.5.1 Navrhované řešení

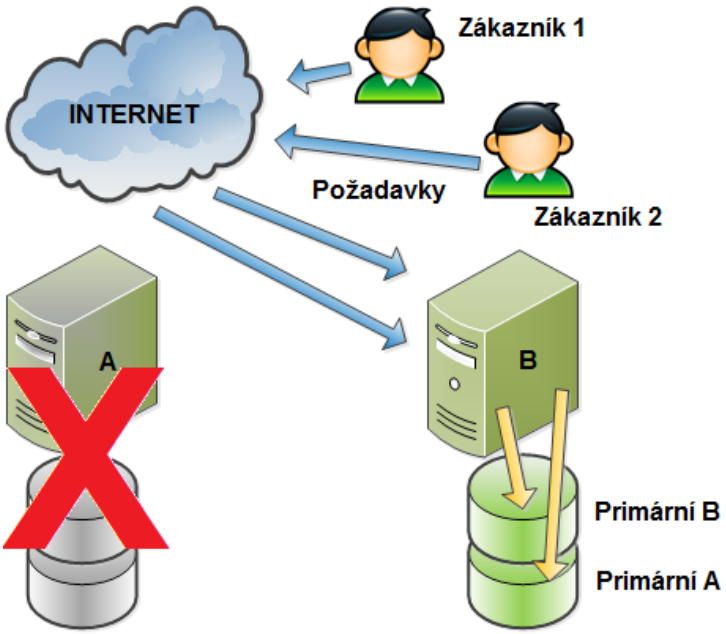
Definovaná kritéria přinášejí jeden složitější pozadavek. Je vyžadováno, aby se oba uzly z clusterové dvojice aktivně účastnily poskytování služby, ale zároveň nelze použít clusterový souborový systém, což znemožňuje použití sdíleného oddílu se společnými daty.

Navrhl jsem tedy takové řešení, že každý uzel clusteru bude mít mít vyhrazen svůj datový oddíl se standardním souborovým systémem, a ten bude přes technologii DRBD v režimu Primary/Secondary synchronizován s protějším uzlem. Princip návrhu je na obrázku 6.4 Každý ze server obsahuje dva diskové oddíly, tedy dvě DRBD zařízení. Jedno zařízení používá v primárním režimu pro svá data, na druhé se mu synchronizují data z protějšího uzlu. Datové disky se tedy synchronizují do kříže mezi uzly.



Obrázek 6.4 Navržené řešení HA clusteru.

Výpadek uzlu pak demonstruje obrázek 6.5.



Obrázek 6.5 Výpadek uzlu..

V případě výpadku uzlu si druhý uzel převezme IP adresu toho havarovaného, prohlásí diskové zařízení s jeho daty za primární a spustí z něj zákaznické kontejnery.

Toto řešení je dobře funkční, přináší jen dva drobnější problémy. Jeden je v neefektivním využití diskové kapacity serveru. Disk je potřeba rozdělit na 2 stejné díly, každý pro jeden z uzlů. V případě sdíleného oddílu by bylo možné použít disk celý. Druhý problém je, že je potřeba rozlišovat, ze kterého serveru, který kontejner pochází, jelikož se fyzicky nalézají na různých diskových částech. Tento problém však vyřeší vhodné nastavení cest v konfiguraci kontejnerů.

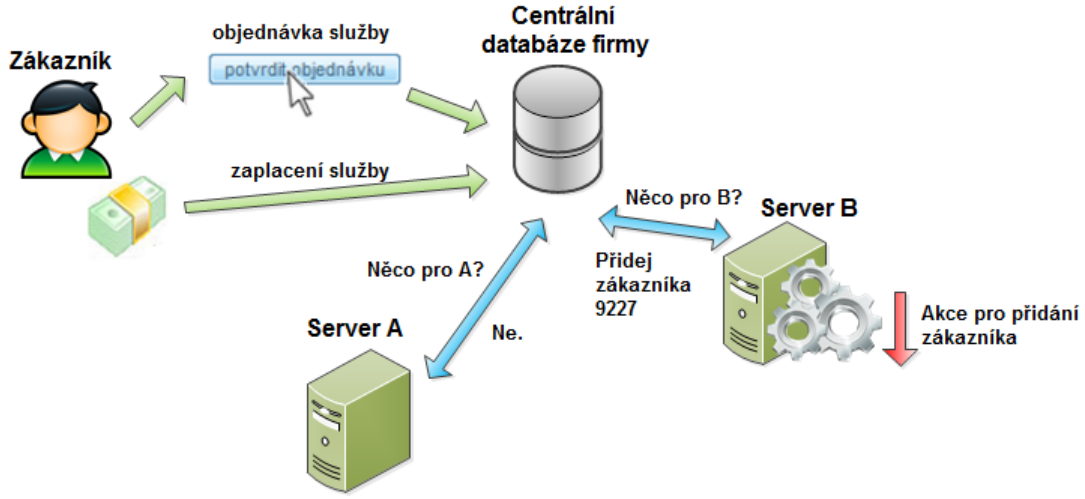
7 Implementace multiwebhostingové platformy

S vybranou virtualizační technologií, vhodným prostředí pro jednotlivé zákaznické kontejnery a návrhem HA clusteru pro zajištění dostupnosti služby jsem mohl přikročit k implementaci kompletní multiwebhostingové platformy na fyzických prostředcích zadavatele.

V této kapitole popíšu postup implementace. Zmíním zde návrh síťové topologie, který kromě virtualizačních hostitelů obsahuje reverzní proxy servery, začlenění hostitelského prostředí kontejnerů a také popíšu automatizaci základních administračních procesů nad multiwebhostingovou platformou.

7.1 Administrační procesy

Než přikročím k popisu implementace samotné multiwebhostingové platformy, kráce se zde ještě zmíním o tom, jak budou nad touto platformou vykonávány administrační procesy v prostředí firmy zadavatele. Firemní prostředí je nastaveno tak, aby ze strany obsluhy firmy bylo potřeba řešit co nejeméně administračních úkonů. Je zde tedy snaha zautomatizovat různé procesy. Na obrázku 7.1 je uveden příklad pro objednávku služby. Zákazník si objedná nějakou službu a zaplatí ji. Servery, které službu realizují si pak v databázi firmy zjišťují, zda pro ně neexistuje nějaká nová akce. Server B například zjistil, že má přidat nového zákazníka. Spustí se tedy vhodný skript, který toto na základě údajů obdržených z databáze realizuje.

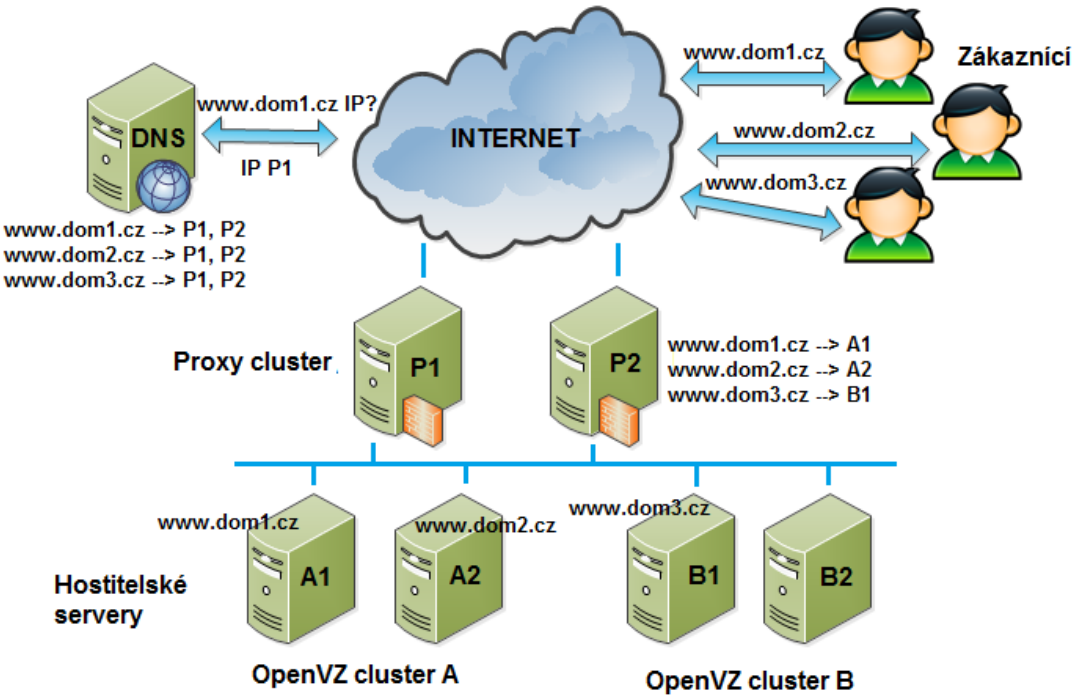


Obrázek 7.1 Administrační procesy.

V zájmu zachování této filosofie jsem se tedy musel zabývat automatizací administračních procesů i na platformě navrhovaného multiwebhostingu.

7.2 Síťová topologie

Pro návrh síťové topologie multiwebhostingové platformy bylo potřeba vzít do úvahy skutečnost, že zde bude spousta zákaznických kontejnerů na jednotlivých hostitelských serverech a každý bude mít pro komunikaci po síti svou vyhrazenou IP adresu. V době, kdy je IP adres verze 4 těžký nedostatek a IP verze 6 se zatím prostě neprosadilo, bylo potřeba přijít s řešením, které pomůže uspořit veřejné IP adresy. Jelikož se jedná o webovou službu a protokol HTTP, je velice snadné využít reverzního proxy serveru, který umožní schovat veliké množství dílčích webových serverů za jednu nebo několik veřejných IP adres.



Obrázek 7.2 Zjednodušení síťová topologie multiwebhostingu.

Na obrázku 7.1 je pak zjednodušený nákres navržené topologie s reverzními proxy servery. Zákazníci a návštěvnící jejich webových stránek k nim přistupují standardně přes názvy domén. DNS systém se stará o mapování těchto adres na IP adresy konkrétních serverů. DNS záznamy pro hostované domény se tedy nastaví na firemní proxy servery a ty znají zase mapování vnitřní sítě a ví, která doména se nalézá na které IP adrese kontejnerů na hostitelích.

Úplná realizace síťové struktury je pak trochu složitější, jelikož servery, které realizují hostitelské prostředí zákaznických kontejnerů potřebují být připojeny do více sítí za různými účely.

- Síť pro komunikaci s proxy servery.
    - Zde probíhá HTTP provoz s webhostingy jednotlivých zákazníků.

- Síť pro komunikaci s dalšími službami.
    - Administrační procesy potřebují zjišťovat například přidání nových kontejnerů nebo úpravu konfigurace webového serveru zákazníka, je tedy nutné spojení s centrální databází firmy. Dále je potřeba síť s poštovními servery.
- Síť pro vlastní připojení hostitelů do internetu
    - Hostitelské servery potřebují vlastní internetovou konektivitu, musí se aktualizovat a spravovat na dálku.
- Síť pro synchronizaci uzlů clusterů.
    - DRBD vyžaduje pro synchronizaci blokových zařízení síťové propojení uzlů.
- Síť pro zálohování.
    - Je potřeba zálohovat zákaznické kontejnery i vlastní konfigurace hostitelských serverů.

V serverové farmě zadavatele jsou k dispozici 2 fyzické sítě s přislušnými servery:

- Vnější síť.
    - Připojení do internetu.
- Vniřní síť.
    - Komunikace s centrální databází (databázové servery).
    - Zálohování (zálohovací servery).
    - Komunikace s poštovními servery (poštovní servery).

Musel jsem tedy ještě vyřešit fyzickou síť pro komunikaci s proxy servery a síť pro synchronizaci uzlů hostitelských clusterů. Z důvodu výkonu jsem se pro síť s proxy servery rozhodl navrhnout zadavateli zavedení další fyzické sítě s vlastní adresací. Pro synchronizaci uzlů clusterů jsem navrhl využívat sítě pro připojení k internetu, jelikož hostitelské servery ji prakticky nebudou používat (veškerý jejich provoz, resp. provoz hostovaných kontejnerů bude probíhat na síti s reverzními proxy servery). Neveřejný IP rozsah pro clustrovou synchronizaci a vhodné nastavení firewallů pak i na společné síťové kartě zajistí dostatečnou míru zabezpečení.

7.3 Reverzní proxy server

Použití HTTP reverzních proxy serverů je nezbytné, jelikož hostitelé multiwebhostingu budou obsahovat velké množství kontejnerů s vlastními IP adresami a není možné použít z úsporných důvodu veřejnou adresaci. Pro reverzní proxy server jsem se rozhodl využít software nginx [34]. Má jednoduchou konfiguraci, je považován za jednu z nejvýkonnějších proxy aplikací vůbec a poskytuje

velice kvalitní dokumentaci. Nebyl tedy problém si dohledat, nastudovat a navrhnout vhdonou konfiguraci.

Reverzní proxy server funguje jako prostředník mezi velkým množstvím klientů a několika servery. Ve své konfiguraci má specifikováno, která doména je obsluhována kterým webovým serverem. Klienti tedy vznášejí požadavky na různé domény, které jsou v systému DNS směrovány na proxy server a ten dle konfigurace přeposílá tyto požadavky na vnitřní webové servery. Ty požadavky vyřídí a zpět přes proxy server vrátí klientům odpovědi. Jako důsledek této funkce je také skrytí obslužných webových serverů za proxy servery, což zvyšuje bezpečnost. Na obrázku 7.3 je pak uveden příklad konkrétní konfigurace nginx pro doménu 101.cz. Tato doména a jakákoliv její subdoména jsou směrovány na obslužný server 172.17.0.5.

```
server {
  listen                        80;
  server_name                   *.101.cz 101.cz;
  location / {
    proxy_pass                  http://172.17.0.5:80;
  }
}
```

Obrázek 7.3 Příklad konfigurace nginx.

7.3.1 Rozkládání zátěže a zajištění dostupnosti

Jelikož je třeba i u proxy serveru řešit zajištění dostupnosti jeho služeb, musí být proxy servery minimálně dva. S tím mě napadla možnost i jednoduchého rozkládání zátěže HTTP provozu. Každé doméně lze v DNS systému nastavit více A záznamů, tedy směrování na konkrétní IP adresu. DNS server pak používá pro dotazy na IP adresu postupně všechny A záznamy tak, jak přichází požadavky. Pokud u zákaznických domén tedy bude vždy uvedena IP adresa obou proxy serverů, mohou se oba proxy servery účastnit příchozího provozu všech domén. To zajišťuje jednoduché rozložení zátěže a ze strany firmy se nemusí řešit, na který proxy server se umístí konfigurace které domény.

Samozřejmě tak jako v případě hostitelských serverů pro multiwebhostingové kontejnery jsem se i v případě reverzního proxy serveru musel zabývat otázkou zajištění dostupnosti jeho služby. Rozhodl jsem se pro jiný model než v případě hostitelských serveru multiwebhostingu. Použil jsem rovněž dva servery, zapojené do pomyslného proxy clusteru. Jelikož vlivem výše popsaného rozhodnutí rozkládat HTTP zátěž na úrovni DNS musí mít oba proxy servery kompletní konfiguraci pro všechny domény, aby kterýkoliv z nich mohl obsloužit libovolnou doménu, stačí tedy vlastně jen zajistit, aby v případě výpadku jednoho z proxy serverů si druhý převzal jeho IP adresu. Požadavky pak budou chodit stále na dvě různé IP adresy dle DNS, ale obě tyto adresy budou nastaveny na funkčním uzlu. Přiřazení IP adresy lze řešit ručně nebo lépe zautomatizovat jej. Proxy servery by se pak musely navzájem nějakým mechanismem prověřovat. Například nejjednodušší řešení založené na

vzájemném propingávání. V případě určité ztráty paketů by byl druhý uzel prohlášen za nedostupný. Elegantnější metoda by pak mohla spočívat přímo v testování dostupnosti HTTP portu.

7.3.2 HTTPS na reverzním proxy serveru

S provozem HTTP souvisí také jeho šifrovaná varianta HTTPS. Z principu není možné do šifrovaného provozu nahlížet, reverzní proxy server by tak nemohl zjistit, pro jakou doménu je HTTP požadavek určen. Proto jsem nakonfiguroval proxy server jako tzv. HTTPS terminátor. Ten naslouchá na patřičném portu (HTTPS – TCP 443), rozšifruje HTTPS spojení od klienta, podívá se pro jakou doménu je provoz určen a přepošle jej na obslužný webový server. To má navíc výhodu, že síťový provoz není potřeba znovu zašifrovat. Obslužné servery se tedy již manipulací s HTTPS nemusí zabývat. Nevýhoda je, že certifikáty pro šifrování musí být umístěny přímo na reverzních proxy serverech.

```
server {
  listen                        443 ssl;
  server_name                   *.101.cz 101.cz;
  ssl_certificate                ssl/default/server.crt;
  ssl_certificate_key            ssl/default/server.key;
  ssl_client_certificate         ssl/default/server.ca;
  location / {
    proxy_pass                  http://172.17.0.5:80;
  }
```

Obrázek 7.4 Příklad konfigurace nginx pro HTTPS.

Na obrázku 7.4 je příklad konfigurace pro HTTPS. Konfigurace se liší od té pro HTTP z obrázku 7.3 pouze ve specifikaci použitého SSL certifikátu.

7.3.3 Konfigurační struktura reverzního proxy serveru

Protože se počítá s automatizací administračních procesů správy konfigurace proxy serverů, musel jsem navrhnout vhodnou strukturu konfiguračních souborů, kterou bude snadné obsluhovat skripty.

```
include                     /etc/nginx/accounts/*/*.conf;
```

Obrázek 7.4 konfigurace domén pro jednotlivé zákazníky v nginxu.

Na obrázku 7.4 je příslušná konfigurace nginx. V */etc/nginx/accounts* jsou umístěny adresáře dle čísel zákaznických účtů. V každém z těchto adresářů jsou pak soubory, kde každý soubor odpovídá definici jedné domény. Pokud je potřeba například přidat novou doménu, stačí jen do adresáře konkrétního zákazníka zapsat nový soubor a restartovat proxy server. Pro ukládání případných SSL certifikátů je použita stejná strategie. Rovněž zrušení domény je jednoduché, jen se umaže

odpovídající definiční soubor. V případě rušení celého zákaznického účtu se smaže příslušný zákaznický adresář se všemi konfiguracemi pro domény.

7.3.4 Automatizace administračních procesů reverzního proxy serveru

Reverzní proxy servery potřebují mít ve své konfiguraci zaneseno jaké doméně odpovídá která IP adresa obsluhujícího kontejneru s webovým serverem. Tato konfigurace se tedy bude měnit podle toho, jak si zákazníci budou přidávat domény na svém multiwebhostingovém účtě. Ruční změna konfigurace reverzních proxy serverů pro velké množství domén a časté požadavky na jejich přidávání je náročná na čas technika ze strany firmy zadavatele. Navíc nemusí proběhnout vždy ihned po zadání požadavku zákazníkem (až se k tomu technik dostane). Proto je požadováno, aby přidávání, resp. rušení domén do konfigurace reverzních proxy serverů probíhalo automaticky.

Zákazník si pohodlně kdykoliv ve webové administraci přidá novou doménu do svého multiwebhostingu. Parametry této akce se pak uloží v centrální databázi firmy. Proxy server tedy obsahuje pravidelně spouštěný skript, který zajišťuje přidání domény do konfigurace proxy serveru. Tento skript je uveden v příloze A.2. Skript si v databázi zjišťuje, existuje-li nová doména k přidání. Pokud ano, provede zápis její konfigurace do struktury specifikované v kapitole 7.3.3. IP adresu obslužného webového serveru pro konfigurační soubor si zjistí z interní DNS a restartuje proxy server.

Pokud zákazník k doméně nahraje i svůj vlastní SSL certifikát, je potřeba v reverzním proxy serveru přidat konfiguraci i pro něj. Zde jsem narazil na jeden menší problém, a sice že pokud zákazník použije certifikát chráněný heslem, proxy server se při načítání nové konfigurace domény zastaví a čeká na toto heslo. Situaci jsem ošetřil převedením zaheslovaných certifikátů na certifikáty bez hesla. Skript pro přidávání domény s vlastním certifikátem je v příloze A.3.

Přidávání konfigurace do proxy serveru provádím vždy jen na jednom z nich (*master*). Na druhý proxy server (*slave*) je pak nová konfigurace nakopírována přes *rsync*, viz skript A.4 v příloze. Slave server pak rozpoznává, že došlo k úpravě konfigurace na základě příznaku, který mu tam master při provedení rsync zanechá v podobě určitého souboru. Pokud slave server nalezne nakopírovaný příznak nové konfigurace, provede restart nginx démona, který realizuje reverzní proxy server, viz skript A.5 v příloze a smaže soubor sloužící jako příznak.

7.4 OpenVZ hostitelský server

Multiwebhosting založený na kontejnerové virtualizaci potřebuje své hostitelské prostředí. Instalace hostitelských serverů je ovlivněna požadavky na implementaci HA clusteru a síťovou topologií. Musí zde být instalováno a patřičně nakonfigurováno vybrané virtualizační řešení, pod kterým se budou jednotlivé kontejnery spouštět. Instalaci hostitele lze rozdělit do několika etap.

- Rozdělení dostupného diskového prostoru.
- Instalace operačního systému Gentoo GNU/Linux.

- Instalace HA clusteru na technologii DRBD.
- Instalace virtualizačního řešení OpenVZ.
- Instalace hostovaného prostředí kontejnerů.
- Instalace FTP serveru na hostitele.
- Automatizace administračních procesů.

7.4.1 Disková struktura hostitele

Hostitelské servery mají k dispozici 2 oddělená disková pole RAID typu 1. Jedno pole slouží pro operační systém hostitele a samotné kontejnerové prostředí virtualizační platformy. Druhé pole jsem vyhradil pro soubory databázových serverů zákaznických kontejnerů.

Pro vlastní operační systém hostitele je potřeba minimálně dvou oddílů. Na jednom bude umístěn odkládací oddíl (*swap*) a na druhém samotný operační systém (*root /*). Je dobré přidat i třetí samostatný oddíl, kde budou uloženy například šablony kontejnerů a další data nesouvisející přímo s hostitelským operačním systémem. Pro hostované prostředí zákaznických kontejnerů jsou pak potřeba dva oddíly dle implementace HA clusteru. Na jednom oddíle bude mít hostitel uloženy kontejnery, které bude obsluhovat a na druhý oddíl se mu budou synchronizovat kontejnery protějšího uzlu pro případ výpadku. Ve stejném duchu je potřeba dvou oddílů na druhém diskovém poli pro databázové soubory. Opět jeden z oddílů bude sloužit pro databáze kontejnerů hostitelského uzlu a druhý pro synchronizaci těchto databází z uzlu druhého v případě jeho výpadku.

7.4.2 Virtualizační řešení OpenVZ

Na server s instalovaným operačním systémem a zařazený do HA clusteru je dále nutné instalovat samotný virtualizační software. To sestává z instalace vhodného jádra a správcovských nástrojů OpenVZ.

Jádro musí být použito speciálně patchované, které dodávají tvůrci virtualizačního řešení. Zde jsem narazil na menší problém, a sice že jádro je dodáváno s obecnou konfiguraci pro sestavení, která zahrnuji téměř vše, co lze v jádře najít a to není žádoucí. Bohužel tím, že jádro je dosti upraveno pro provoz kontejnerové virtualizace, není jeho překonfigurování tak snadné. Pokud změníme některé volby, následně není možné jádro přeložit. Nechceme-li tedy mít jádro, které zahrnuje veškerý dostupný obsah, je potřeba postupně odebírat nechtěné moduly a zkoušet jádro překládat. V případě, že jádro nelze přeložit, musíme součásti odebrané v posledním kroku navrátit zpět.

Další omezení, na která jsem narazil při instalaci OpenVZ RHEL6 jádra jsou uvedena níže:

- Lze použít binutils maximálně ve verzi 2.20.1-r1, jinak jádro nelze přeložit.
- Lze použít gcc maximálně verze 4.4.5, jinak jádro nelze nabootovat.

- Lze použít linux – headers maximálně verze 2.6.36.1, jinakl jádro nelze přeložit.

Správcovské nástroje OpenVZ jsou dostupné přímo v instalačním stromu portage a s jejich instalací není žádný problém. Je potřeba pouze drobně upravit konfiguraci sítě.

OpenVZ standardně používá pro odchozí síťový provoz kontejnerů fyzické síťové rozhraní hostitele, které svou adresací odpovídá adresaci rozhraní v kontejneru. Jelikož je kontejner adresován pro použití v síti s reverzními proxy servery, snaží se tedy OpenVZ veškerou komunikaci směrovat na toto síťové rozhraní. Kontejnery však potřebují přistupovat i ke službám z dalších sítí hostitele, například k mailovým serverům. Tato síť je však na hostiteli připojena k jinému fyzickému rozhraní. Aby nebylo nutné nastavovat v kontejnerech více IP adres, tak jsem v konfiguraci OpenVZ musel upravit volbu *NEIGHBOUR_DEVS* na hodnotu *all*. To zapříčiní, že odchozí požadavky kontejnerů nebudou předávány jen na fyzické rozhraní hostitele se stejnou adresací.

7.4.3 Kontejnerové prostředí OpenVZ

Kontejnerové prostředí sestává ze sdílené šablony operačního systému a privátního prostoru pro jednotlivé kontejnery tak, jak jsem provedl návrh v kapitole 4.

Na disku je vyhrazena určitá administrační oblast, kde je uložena výchozí privátní struktura kontejneru, prázdná instalace databázového serveru a sdílená šablona. Při každém přidávání kontejneru je provedeno nakopírování výchozí privátní struktury a oblasti s databázovým serverem. Po spuštění kontejneru se do zákaznické privátní struktury připojí obraz se šablonou a databázovými soubory.

7.4.4 Automatizace administračních procesů hostitele

Pro okamžité nasazení jsem musel jsem zautomatizovat dva administrační procesy. Zakládání kontejnerů pro nové zákazníky a úpravu konfigurace webových serverů těchto kontejnerů. Skripty, které toto realizují, jsou pravidelně spouštěny plánovačném úloh (*cron*) a dle údajů v databázi provedou požadovanou akci.

Příklad skriptu pro přidávání nového kontejneru je uveden v příloze A.6. Pokud je požadováno přidání nového kontejneru, skript nejprve nakopíruje výchozí struktury, zapíše konfiguraci kontejneru, která obsahuje především omezení systémových zdrojů a spustí kontejner. Zpětně v databázi označí, že kontejner byl vytvořen a je připraven k použití, tím se klientovi aktivuje webová administrace jeho multiwebhostingového účtu. Skript pro úpravu konfigurace webového serveru jsem převzal ze sdíleného webhostingu, kde stačilo jen upravit cesty ke konfiguraci webového serveru dle zákaznických kontejnerů.

8 Testování a monitoring multiwebhostingové platformy

Dokončenou implementaci multiwebhostingové platformy jsem musel dále opatřit monitorovacím subsystémem a platformu také otestovat, než bude uvedena do produkčního prostředí.

8.1 Implementace monitorování kontejnerů

Monitorování kontejnerů je mnohostranně výhodné. Firma, která službu provozuje i zákazník si mohou ověřit, v jakém stavu se kontejner nalézá a jaké spotřebovává systémové zdroje. Pro monitorování systémových zdrojů jednotlivých kontejnerů jsem využil přímo speciálních struktur OpenVZ jádra, ze kterých je možno pro jednotlivé kontejnery zjišťovat využití zdrojů. Tyto struktury jsou připojeny do */proc*.

Zjišťuje se:

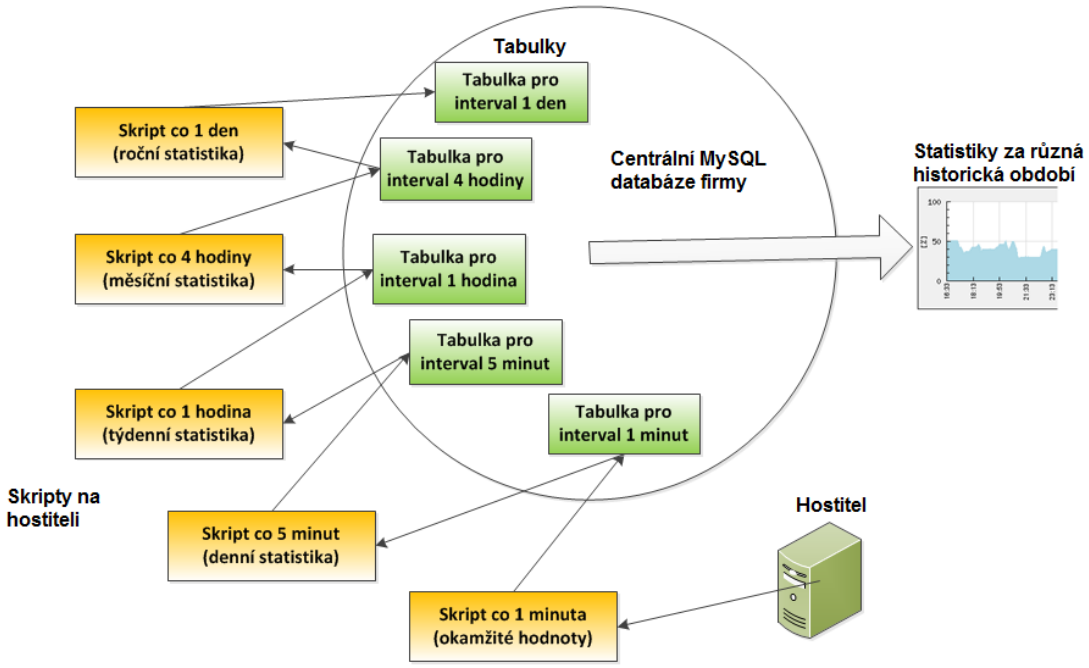
- Zátěž procesoru CPU.
- Obsazení RAM.
- Celkový počet procesů.
- Zabrané místo na disku.
- Počet obsazených inodů.
- Počet TCP a ne – TCP socketů.

Na základě požadavku firmy jsem ještě dodal také statistiky týkající se MySQL serveru. Tyto údaje zjišťuji připojením se k databázi kontejneru a dotázáním se na patřičnou hodnotu.

Zjišťuje se:

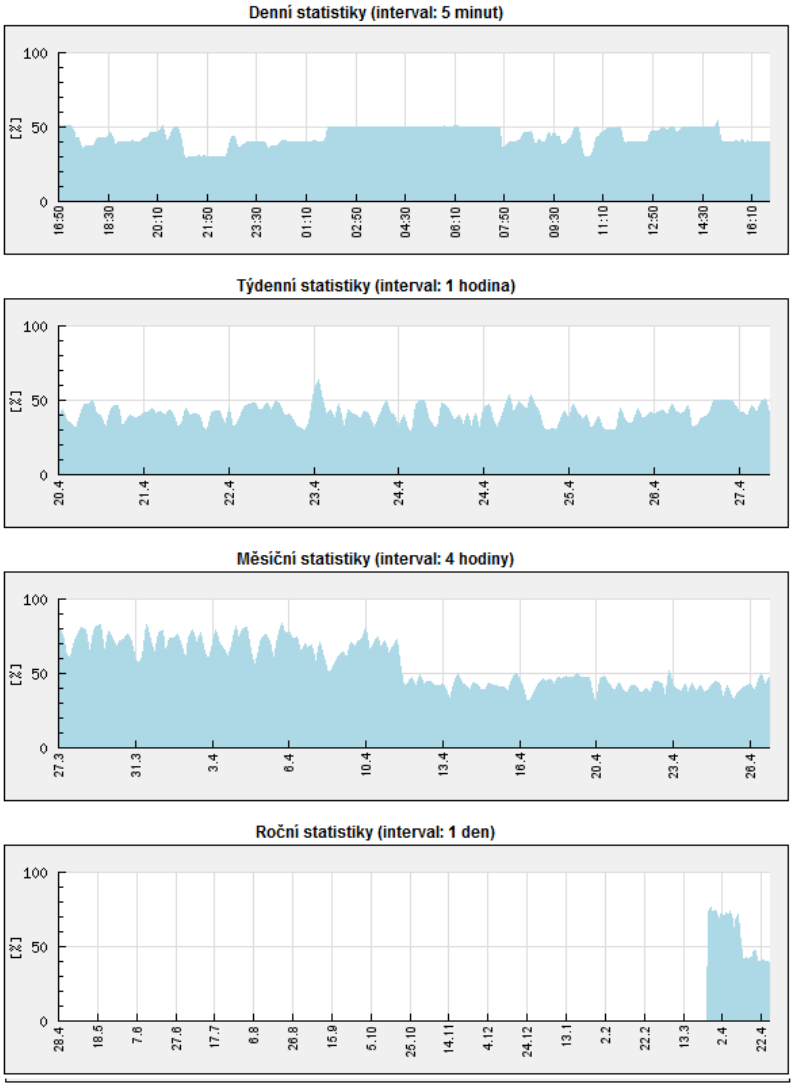
- Počet současných spojení na DB server.
- Počet dotazů za vteřinu.

Skript, který zjistí aktuální využití zdrojů jednotlivými kontejnery tato data uloží do MySQL databáze každou minutu. Aby bylo možno poskytovat graficky zobrazenou historii sledovaných zdrojů je potřeba dále tato data přepočítávat za různá období po různých intervalech.



Obrázek 8.1 Statistiky za různá historická období.

Na obrázku 8.1 je uveden princip návrhu monitorování za různá historická období. Na hostiteli je každou minutu volán skript, který uloží data okamžitých hodnot systémových zdrojů zabraných kontejnery. Další skripty spouštěné v určitých intervalech provádějí vždy průměrový přepočet dat sebraných za předchozí časové období. Skript pro roční statistiku tak přepočítává data z tabulky pro měsíční statistiku, skript pro měsíční statistiku zase přepočítíví data z tabulky pro týdenní statistiku a tak dále. Výsledky jsou pak vykreslovány každému zákazníkovi v administraci jeho multiwebhostingového účtu. Viz obrázek 8.2.



Obrázek 8.2 Statistika využití operační paměti v administraci multiwebhostingu.

Skript pro získávání hodnot využití systémových zdrojů všech kontejnerů je uveden v příloze A.7.

8.2 Testování multiwebhostingové platformy

Jelikož každý z kontejnerů má jistou režii a svůj vlastní webový a databázový server, nelze očekávat, že na jednoho hostitele bude možné umístit stejný počet zákazníků jako při sdíleném webhostingu. To je však vyváženo širší možností konfigurace a mírně vyšší cenou. Očekává se, že jeden hostitel se čtyřjádrovým procesorem a 24 GB operační paměti by měl být schopen obsloužit 300 až 400 zákaznických kontejnerů. Na jeden kontejner by tedy vycházelo při plném využití operační paměti 64 MB RAM. Počítá se však s tím, že všechny kontejnery naráz nebudou požadovat plnou výši

uvedených zdrojů a proto je možno nabízet limity operační paměti vyšší. Stejný případ je i u ostatních systémových zdrojů.

Při testování stability jsem se snažil zatížit kontejnery s instalovanými redakčními systémy Wordpress velkým množstvím příchozích požadavků. Pokud se kontejner dostane do limitace operační paměti, není dovoleno vytvoření dalších procesů, případně jsou zabíjeny aktuálně spuštěné procesy. Na straně klienta se to pak projeví nenačtením stránek, případně nějakým chybovým kódem webového serveru. Celý hostitelský uzel se mi přetížit nepovedlo. OpenVZ chrání hostitelský počítač tím, že pro něj alokuje minimální systémové prostředky, které nedovolí uvolnit pro potřeby kontejnerů.

Drobný problém jsem zaznamenal při implementaci skriptů pro automatizaci procesů přidávání domén do reverzních proxy serverů. Kompletní skript je uveden v příloze A.2. Skript si načte z databáze novou konfiguraci, ověří několik podmínek (dostupnost všech požadovaných parametrů na zanesení konfigurace, zda už konfigurace domény neexistuje, zda DNS vrací IP adresu obslužného serveru, apod.) a jsou-li vpořádku, přidá konfiguraci. Pokud vpořádku nejsou, není konfigurace domény zapsána. Bohužel toto jsem zpětně nesignalizoval do databáze a při dalším periodickém spouštění skriptu se skript opět snažil tuto doménu přidat a opět se stejným výsledkem. Tato chyba pak byla s každým spuštěním skriptu hlášena e-mailem jako chyba.

V současnosti je již platforma v produkční kvalitě, není však oficiálně nabízena jako placená. Firma se snaží přilákat větší množství zákazníků ať již nových nebo stávajících, aby se platforma otestovala v běžném produkčním provozu. S klidem musím prohlásit, že prozatím se neprojevil žádný neočekávaný nedostatek.

9 Závěr

Pro firmu SvetHostingu.cz jsem provedl realizaci multiwebhostingové platformy. Řešení nabízí zákazníkům webhostingový prostor s maximální konfigurovatelností a vlastními vyhrazenými systémovi zdroji, které však efektivně a dynamicky využívá místo vyhrazení maximální možné přidělené hranice. Do tohoto řešení jsem také implementoval technologii HA clusteru pro zajištění dostupnosti služby. Havárie hostitelského serveru tak neznamená dlouhý výpadek, který se dotkne mnoha zákazníků. Služba je dále ze strany firmy snadno spravovatelná. Pro kontejnery jsem navrhl sdílený režim, který umožňuje využít pouze jedné společné šablony a odděluje pouze zákaznická data a specifické konfigurace. Dle filosofie firmy provádět procesy automaticky jsem také zautomatizoval většinu administračních procesů nezbytných pro uvedení služby. Vytvoření nového kontejneru a úprava konfigurace webového serveru tak probíhá zcela automaticky pouze na základě údajů zanesených do centrální databáze firmy webovým administračním rozhraním pro zákazníky.

Z hlediska budoucího rozvoje platformy zůstala otevřena otázka IP adresace verze 6, která nebyla v současnosti implementována, ale navržená struktura je na ni plně připravena. Dále bude vhodné pro platformu doplnit automatické řešení výpadků. V současnosti je nutné ručně připojit oddíl havarovaného hostitele a spustit jeho kontejnery. Rovněž není zautomatizován proces převzetí IP adresy při výpadku proxy serveru. Lze také zautomatizovat ještě další administrační procesy, jejichž akce zatím nebyly potřeba, například rušení kontejneru.

10 Literatura

[1] DUC, Václav. *Absolvování individuální odborné praxe*. Vysoká škola báňská - Technická univerzita Ostrava, 2010. Dostupné z: http://hdl.handle.net/10084/78794. Bakalářská práce. Vysoká škola báňská - Technická univerzita Ostrava. Vedoucí práce Eduard Sojka.

[2] RFC 2616: Hypertext Transfer Protocol -- HTTP/1.1. The Internet Engineering Task Force [online]. 1999 [cit. 2012-05-02]. Dostupné z: http://www.ietf.org/rfc/rfc2616.txt

[3] Name-based Virtual Host Support: Apache HTTP Server Version 2.0. The Apache Software Foundation [online]. 2011 [cit. 2012-05-02]. Dostupné z: http://httpd.apache.org/docs/2.0/vhosts/name-based.html

[4] Chroot. Wikipedia, the free encyclopedia [online]. 2012 [cit. 2012-05-02]. Dostupné z: http://en.wikipedia.org/wiki/Chroot

[5] Kernel Based Virtual Machine. Kernel Based Virtual Machine [online]. 2012 [cit. 2012-05-02]. Dostupné z: http://www.linux-kvm.org/page/Main_Page

[6] Virtio. Libvirt: The virtualization API [online]. 2012 [cit. 2012-05-02]. Dostupné z: http://wiki.libvirt.org/page/Virtio

[7] Management Tools - KVM. Kernel Based Virtual Machine [online]. 2012 [cit. 2012-05-02]. Dostupné z: http://www.linux-kvm.org/page/Management_Tools

[8] Xen.org. Xen.org [online]. 2012 [cit. 2012-05-02]. Dostupné z: http://www.xen.org/

[9] Virtual Machine Manager. Virtual Machine Manager [online]. 2012 [cit. 2012-05-02]. Dostupné z: http://virt-manager.et.redhat.com/

[10] Linux-VServer.org. Linux-VServer.org [online]. 2012 [cit. 2012-05-02]. Dostupné z: http://linux-vserver.org/Welcome_to_Linux-VServer.org

[11] OpenVZ Linux Containers. OpenVZ Linux Containers [online]. 2012 [cit. 2012-05-02]. Dostupné z: http://wiki.openvz.org/Main_Page

[12] Parallels. Virtualization & Automation Solutions for Desktops, Servers, Hosting, SaaS – Parallels Optimized Computing [online]. 2012 [cit. 2012-05-02]. Dostupné z: http://www.parallels.com

[13] OS Virtualization Solution for Windows and Linux — Parallels Virtuozzo Containers. Virtualization & Automation Solutions for Desktops, Servers, Hosting, SaaS – Parallels Optimized Computing [online]. 2012 [cit. 2012-05-02]. Dostupné z: http://www.parallels.com/eu/products/pvc/

[14] Control panels for OpenVZ. OpenVZ Linux Containers [online]. 2012 [cit. 2012-05-02]. Dostupné z: http://wiki.openvz.org/Control_panels

[15] Red Hat Enterprise Linux. Red Hat Enterprise Linux [online]. 2012 [cit. 2012-05-02]. Dostupné z: http://www.redhat.com/products/enterprise-linux/

[16] Oracle VM VirtualBox. Oracle VM VirtualBox [online]. 2012 [cit. 2012-05-02]. Dostupné z: https://www.virtualbox.org/

[17] Ab - Apache HTTP server benchmarking tool. The Apache Software Foundation [online]. 2012 [cit. 2012-05-02]. Dostupné z: http://httpd.apache.org/docs/2.0/programs/ab.html

[18] LVM HOWTO. LVM HOWTO [online]. 2006 [cit. 2012-05-02]. Dostupné z: http://tldp.org/HOWTO/LVM-HOWTO/

[19] Gentoo Linux. Gentoo Linux [online]. 2012 [cit. 2012-05-02]. Dostupné z: http://www.gentoo.org/

[20] Gentoo template creation. OpenVZ Linux Containers [online]. 2012 [cit. 2012-05-02]. Dostupné z: http://wiki.openvz.org/Gentoo_template_creation

[21] OpenRC and baselayout 2 will be stabilized on May 8. Gentoo Linux [online]. 2011 [cit. 2012-05-02]. Dostupné z: http://www.gentoo.org/news/20110505-openrc.xml

[22] R57 Shell: http://www.r57.gen.tr/. R57 Shell [online]. 2009 [cit. 2012-05-02].

[23] High-availability cluster. Wikipedia, the free encyclopedia [online]. 2012 [cit. 2012-05-02]. Dostupné z: http://en.wikipedia.org/wiki/High-availability_cluster

[24] Clustered file system. Wikipedia, the free encyclopedia [online]. 2012 [cit. 2012-05-02]. Dostupné z: http://en.wikipedia.org/wiki/Cluster_file_system

[25] Project: OCFS2. Free and Open Source Software from Oracle [online]. 2009 [cit. 2012-05-02]. Dostupné z: http://oss.oracle.com/projects/ocfs2/

[26] GFS Project Page. GFS Project Page [online]. 2012 [cit. 2012-05-02]. Dostupné z: http://sourceware.org/cluster/gfs/

[27] Comparison of file systems. Wikipedia, the free encyclopedia [online]. 2012 [cit. 2012-05-02]. Dostupné z: http://en.wikipedia.org/wiki/Comparison_of_file_systems

[28] ReiserFS. Wikipedia, the free encyclopedia [online]. 2012 [cit. 2012-05-02]. Dostupné z: http://en.wikipedia.org/wiki/ReiserFS

[29] Ext3. Wikipedia, the free encyclopedia [online]. 2012 [cit. 2012-05-02]. Dostupné z: http://en.wikipedia.org/wiki/Ext3

[30] Jumbo frame. Wikipedia, the free encyclopedia [online]. 2012 [cit. 2012-05-02]. Dostupné z: http://en.wikipedia.org/wiki/Jumbo_frame

[31] Bonnie++. Bonnie++ [online]. 2012 [cit. 2012-05-02]. Dostupné z: http://www.coker.com.au/bonnie++/

[32] DRBD. DRBD [online]. 2011 [cit. 2012-05-02]. Dostupné z: http://www.drbd.org/

[33] ISCSI. Wikipedia, the free encyclopedia [online]. 2012 [cit. 2012-05-02]. Dostupné z: http://en.wikipedia.org/wiki/ISCSI

[34] Nginx. Nginx [online]. 2012 [cit. 2012-05-02]. Dostupné z: http://nginx.org/

11 Přílohy

Tyto přílohy jsou umístěny pod stejnými názvy na doprovodném CD u tištěné verze diplomové práce. Z důvodu zachování firemního tajemství nejsou tyto přílohy součástí veřejné verze diplomové práce.

A.1
Skript pro nastavení ACL ve sdílené šabloně.

A.2
Skript pro přidání domény s obecným SSL certifikátem do konfigurace proxy serveru.

A.3
Skript pro přidání domény s vlastním SSL certifikátem do konfigurace proxy serveru.

A.4
Skript pro zkopírování nové konfigurace master proxy serveru na slave.

A.5
Skript pro restart slave serveru, došlo-li ke změně konfigurace masterem.

A.6
Skript pro přidání nového kontejneru.

A.7
Skript pro monitorování kontejnerů.